VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2018 Josef Novotný

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV RADIOELEKTRONIKY
DEPARTMENT OF RADIO ELECTRONICS

DETEKČNÍ ALGORITMY POHYBUJÍCÍCH SE OBJEKTŮ
DETECTION ALGORITHMS OF MOVING OBJECTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Josef Novotný**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Marcoň, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Elektronika a sdělovací technika**
Ústav radioelektroniky

Student: Josef Novotný ***ID:*** 186150
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Detekční algoritmy pohybujících se objektů

POKYNY PRO VYPRACOVÁNÍ:

Proveďte rešerši algoritmů vhodných pro detekci pohybujících se objektů. Nasnímejte video pro testování algoritmů. Video by mělo obsahovat letící objekty, např. hejno ptáků, letadlo, dron. Ve vybraném programu proveďte testování detekčních algoritmů.

Realizujte modul, který bude obsahovat kameru a mikrokontrolér. Do procesoru mikrokontroléru implementujte vybraný detekční algoritmus. Celý modul reálně otestujte a proveďte statistické vyhodnocení získaných údajů.

DOPORUČENÁ LITERATURA:

[1] ZAPLATÍLEK, Karel. MATLAB®: začínáme se signály. Brno: Tribun EU, 2015. Knihovnicka.cz. ISBN 978-80-263-0898-0.

[2] UPTON, Eben a Gareth HALFACREE. Raspberry Pi: uživatelská příručka. 2., aktualizované vydání. Přeložil Jakub GONER. Brno: Computer Press, 2016. ISBN 978-80-251-4819-8.

Termín zadání: 5. 2. 2018 ***Termín odevzdání:*** 24. 5. 2018

Vedoucí práce: Ing. Petr Marcoň, Ph.D.

prof. Ing. Tomáš Kratochvíl, Ph.D.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta elektrotechniky a komunikačních technologií
Ústav radioelektroniky

ABSTRAKT

Tato práce se zabývá metodami detekce pohybujících se objektů. Součástí práce je rešerše dostupných řešení a následná implementace na platformu Raspberry Pi s využitím knihovny OpenCV. Cílem práce je realizovat modul, který je schopný detekovat a rozpoznat pohybující se objekty na obloze. Požadované objekty je možné detekovat jak ve statickém, tak i dynamickém režimu. V teoretické části práce jsou metody děleny na detekci pohybu, objektů a rozpoznávání objektů. V praktické části byla pro detekci nežádoucích objektů zvolena metoda subtrakce pozadí a lokalizace významných bodů ORB, pro rozpoznávání naučených objektů pak kaskádní klasifikátor s Haarovými příznaky. V praktické části byly metody testovány na výpočetní náročnost a úspěšnost detekce. Výsledkem je realizace automatického detekčního zařízení, které umožňuje sledovat a rozlišit nežádoucí objekty na obloze.

KLÍČOVÁ SLOVA

Detekce objektů, Raspberry Pi, OpenCV, Python, ORB, Viola-Jones.

ABSTRACT

This work deals with detection methods of moving objects. This thesis contains a research of available solutions for further implementation on the Raspberry Pi using the OpenCV library. The aim of the thesis is to realize a module that is capable of detecting and recognizing moving objects in the sky. Required objects can be detected in static or dynamic mode. In the theoretical part of the thesis are methods divided into motion detection, object detection and object recognition. In the practical part were for detection of undesired objects selected a background subtraction method and the localization of significant ORB points. Haar feature-based cascade classifiers was for recognition of learned objects chosen. In the practical part, methods were of computational difficulty and detection success tested. The result is an automatic detection device that allows to watch and distinguish unwanted objects in the sky.

KEYWORDS

Object detection, Raspberry Pi, OpenCV, Python, ORB, Viola-Jones.

NOVOTNÝ, Josef. *Detekční algoritmy pohybujících se objektů*. Brno, 2019, 75 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky. Vedoucí práce: Ing. Petr Marcoň, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Detekční algoritmy pohybujících se objektů“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Petru Marcoňovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM VÝPISŮ

ÚVOD

Tato práce se zabývá metodami detekce pohybujících se objektů. Pro tuto práci jsou těmito předměty letadla, ptáci, drony a další objekty nacházející se na obloze.

Před aplikací dále uvedených metod je nezbytné obraz předupravit. Metody pro předzpracování obrazu jsou popsány v kapitole 2 a dále aplikovány v praktické části.

Pro detekci pohybujících se objektů lze využít jejich důležité vlastnosti, kterou je jejich pohyb. Metody založené na sledování pohybujících se segmentů v obraze jsou popsané v kapitole 3. Podmínkou správného vyhodnocení těchto metod je statické upevnění kamery a neustálý pohyb sledovaného objektu.

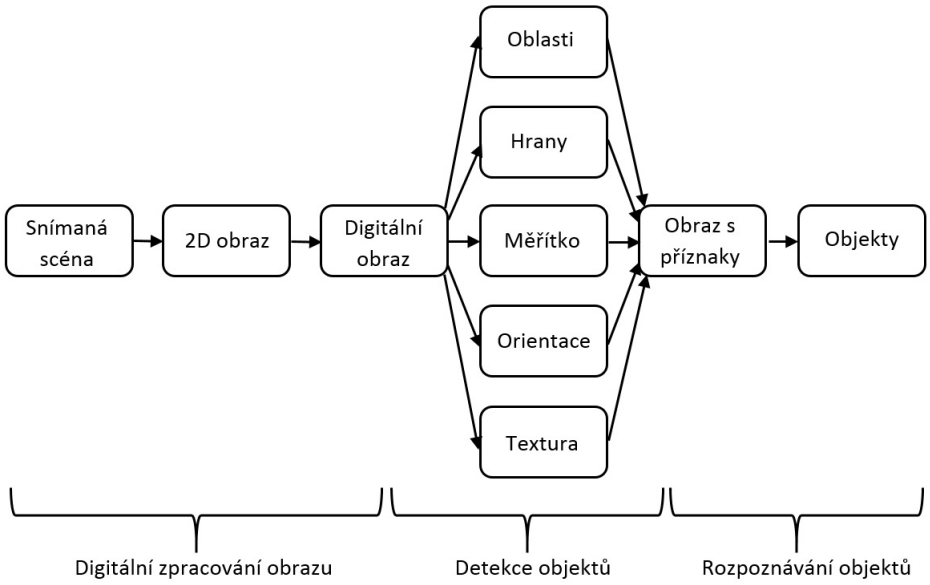
V případě, kdy není možné použít některou z pohybových metod, a to například z důvodu pohybu kamery, pak je vhodné objekt detekovat s pomocí jiného algoritmu. Tyto metody jsou popsány v kapitole 4. Jde o metodu korelační analýzy, lokalizace významných bodů a oblastí. Dalším stupněm je rozpoznávání letadel, či jiných předem definovaných objektů, a to za použití kaskádního klasifikátoru. Příkladem těchto metod je detekce Viola-Jones s Haarovými příznaky.

V předposlední kapitole jsou shrnuty výsledky testování jednotlivých algoritmů a je zde popsána jejich implementace. Pro tuto práci bylo zvoleno Raspberry Pi 3B s otočným modulem, které dokáže rozlišit prolétající letadlo a sledovat jeho dráhu. Dále je možné detekovat nežádoucí objekty, které se vyskytují na obloze. Závěrečná realizace zařízení se nachází v kapitole 6.

1 POČÍTAČOVÉ VIDĚNÍ

Počítačové vidění je obor, který se v posledních letech začal velmi dynamicky rozvíjet, a to především díky nástupu dostatečně výkonné výpočetní techniky. Mezi hlavní cíle tohoto oboru patří schopnost získávat informace z nasnímaného obrazu, následně je zpracovat a výsledně interpretovat. Z hlediska počítačového vidění lze říct, že se snaží různými výpočetními mechanismy vytvořit autonomní systémy, které mohou být srovnatelné a do jisté míry nahraditelné lidským vnímání. Využití takových systémů přináší celou řadu výhod. Mezi největší patří zejména rychlost vyhodnocení, přesnost a široké možnosti využití. Uplatnění těchto systémů nalezneme v oblastech automatizace, lékařské diagnostice a také v kamerových sledovacích zařízeních. [2]

Fáze, které vedou od digitálního zpracování obrazu až po úlohy počítačového vidění, lze popsat obrázkem 1.1 . Na úrovni digitálního zpracování nedochází k žádné interpretaci objektů, v této fázi se pouze zpracovává vstupní dvojrozměrný obraz. Posléze lze obraz analyzovat a detekovat oblasti, hrany či stanovit měřítka a orientace na jejichž základě získáme obraz s příznaky. V závěru lze aplikovat metody počítačového vidění, které dokáží porozumět obsahu snímané scény.

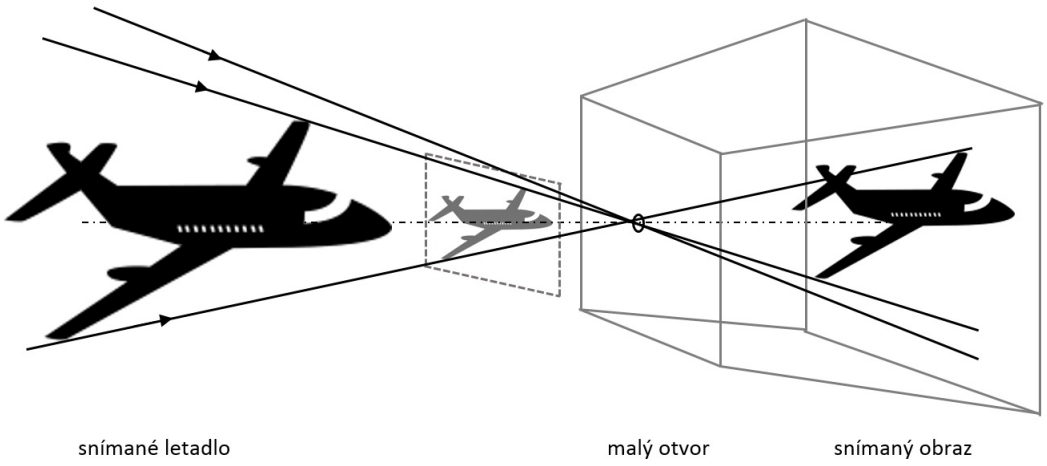


Obr. 1.1: Schéma znázorňující fáze zpracování obrazu na různých úrovních. Inspirace z literatury [1]

1.1 Omezení při zpracování obrazu

Při snímání obrazu a následném zpracování lze narazit na řadu překážek, které jsou omezující pro výsledné zpracování obrazu. Při digitálním zpracování se proto pracuje s obrazovými daty tak, aby byla pro zvolený algoritmus snadno čitelná. Mezi největší omezení patří:

1. **Ztráta informací z transformace 3D obrazu na 2D**
   Prostor, který je snímán kamerou, je převáděn na dvojrozměrný obraz. Jeho geometrické vlastnosti lze popsat tzv. *„pinhole“* modelem, který představuje obr. 1.2. Jedná se o krychli s relativně malým otvorem, kde na zadní stěnu je promítán snímaný obraz. Problém tohoto modelu spočívá v tom, že vzdálené objekty jsou snímány stejně jako blízké předměty bez měřítka. Nelze tak zjistit velikost předmětu ani jeho vzdálenost. [1] Na druhou stranu ztráta informace při transformaci do dvojrozměrného obrazu může přinést usnadnění v mnoha aplikacích, kde je práce s trojrozměrným modelem nadbytečná. Tyto aplikace zahrnují většinu dále zmíněných metod, kde není potřeba zjišťovat vzdálenost či přesný 3D model objektů.



Obr. 1.2: Model ilustrující geometrii, která nerozlišuje velikost objektů. Inspirace z literatury [1]

2. **Interpretace**
   Interpretace obrazu je nepostradatelný úkol počítačové vidění. Lidská schopnost řešit problémy na základě dlouhodobě získaných znalostí je v případě počítačového zpracování omezená. Z matematického pohledu lze obraz logicky interpretovat pomocí vhodně zvoleného modelu. [1]

3. **Šum**
   Šum je ze své podstaty v reálném světě přítomný při každém měření. Můžeme jej jen velmi obtížně popsat matematickými metodami. [1] Nejběžnější odstanění šumu se provádí Gaussovým či mediánovým filtrem, dále zmíněných v kapitole 2.

4. **Objem dat**
   Objem dat přímo souvisí s rozlišením videosnímků. Při pouhé detekci pohybu je nadbytečné mít vysoké rozlišení obrazu, naopak při rozpoznávání vzdálenějších objektů je nezbytné zajistit kvalitnější vstup. Dále je též nutné zahrnout v úvahu, jestli bude aplikace pracovat v reálném čase a případně volit vhodný kompromis.

5. **Měřený jas**
   Jas snímaného obrazu závisí na osvětlení předmětu, na typu světelných zdrojů, jejich poloze, intenzitě, místu pozorovatele, lokální geometrii povrchu a odrazivosti povrchu.

6. **Lokální a globální pohled**
   Záleží na zvoleném přístupu algoritmu. V případě lokální analýzy pracujeme pouze s úzkým výsekem dat. Při popisu celkového obrazu se naopak využívá globální pohled. [1]

Všechna zmíněná omezení je nutné brát v úvahu i dále u jednotlivých metod, které jsou rozebrány v následujících kapitolách.

2 PŘEDZPRACOVÁNÍ OBRAZU

Hlavním úkolem digitálního zpracování obrazu je převod snímku do digitální podoby. Na digitální obraz pak lze aplikovat metody, které vedou ke zlepšení parametrů obrazu nebo výběru nějaké užitečné informace.

Předzpracování obrazu je soubor operací, které se provádí u každého snímku. Cílem těchto operací je upravit vstupní obraz tak, aby jeho následné použití bylo co nejpřesnější a nejrychlejší pro navazující metody [3], popsané v následujících kapitolách 3 a 4.

2.1 Operace s pixely

V této podkapitole jsou uvedeny metody, které upravují vstupní obraz. Pro tyto metody je společné, že každý pixel v obraze je nahrazen jiným o nové hodnotě. Mezi takové metody patří prahování, dilatace či eroze.

2.1.1 Prahování

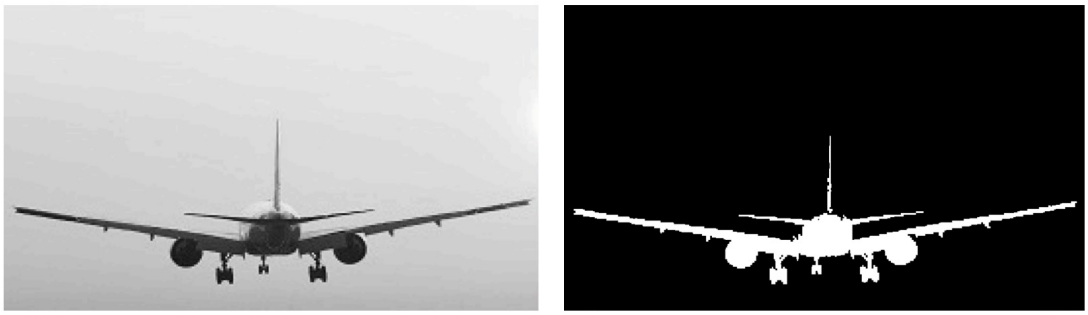
Prahování je jedna ze základních pixelových operací, která vybírá body s pevně stanovenou prahovou hodnotou nebo body nacházející se ve vybraném rozsahu

$$f(x,y) = \begin{cases} 1 & pokud\ T_1 < f(x,y) < T_2\ nebo\ f(x,y) > T, \\ 0 & jinak, \end{cases} \tag{2.1}$$

kde $f(x,y)$ je binární obraz a T je zvolený práh. [3]

Cílem prahování je převod šedotónového snímku do binárního obrazu, viz obr. 2.1. Pixely, nacházející se ve vymezeném rozsahu anebo překračují danou mez, jsou označeny binární hodnotou 1.

Hlavním cílem prahování je oddělit detekovaný objekt od okolního pozadí. Nezbytným předpokladem je proto rozdílná intenzita jasu objektů od pozadí snímku. Pokročilejší metody vybírají hranici prahování automaticky na základě obrazových histogramů. [4]



Obr. 2.1: Nalevo původní šedotónový snímek, napravo prahovaný binární snímek

2.1.2 Vyhlazení průměrováním

Obrazové vyhlazení vzniká konvolucí obrazu s filtrem typu dolní propust. Taková operace je výhodná z hlediska odstranění bílého šumu. Tato operace odstraňuje vyšší frekvence, které představují detaily v obraze, příkladem je pravý snímek na obr. 2.2. Průměrovací operátor K pracuje na předloze o rozměru 3x3 pixely:

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.[5] \tag{2.2}$$

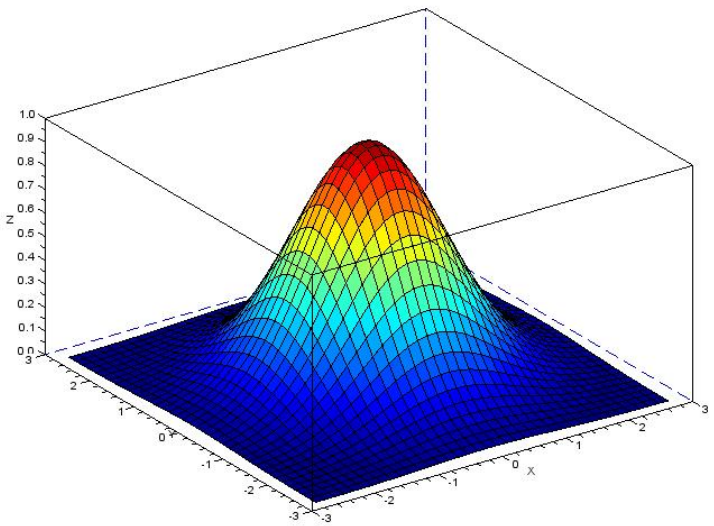
Obr. 2.2: Nalevo původní šedotónový snímek, napravo snímek vzniklý konvolucí s průměrovací maskou.

2.1.3 Gaussovo vyhlazení

V případě Gaussova vyhlazení je místo filtru s konstantními koeficienty použito Gaussovo jádro, jenž je znázorněno na obr. 2.3. Gaussova funkce $g(x, y, \sigma)$ je definována jako

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}, \tag{2.3}$$

kde σ je směrodatná odchylka s jejíž pomocí nastavujeme míru filtrace. Při zvolení malého σ je filtrace nevýrazná, a naopak při vyšších hodnotách dochází k poškození hran. Gaussovo vyhlazení je velmi efektivní pro potlačení Gaussova šumu. [5]



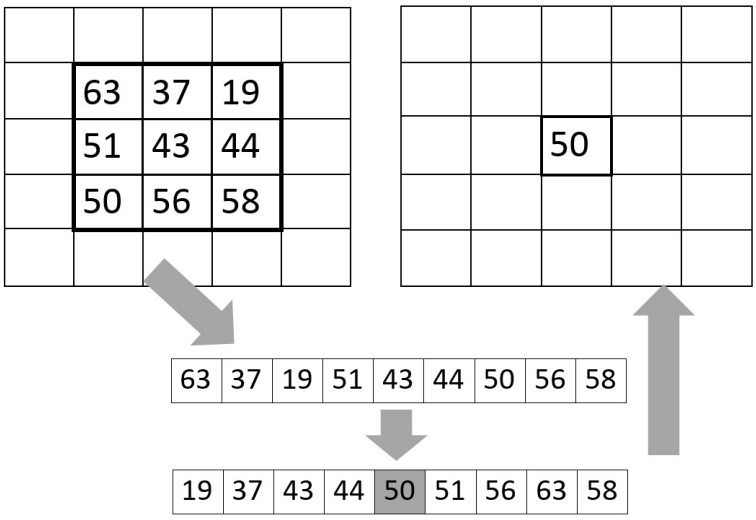
Obr. 2.3: Gaussovo jádro. [6]

Z rovnice 2.3 jsou vypočteny koeficienty pro Gaussovo jádro, které jsou následně použité při konvoluci se vstupním obrazem. Zároveň je nezbytné specifikovat velikost tohoto konvolučního jádra, standardně se používá jádro o velikosti 5x5 pixelů. [4]

2.1.4 Mediánový filtr

Mediánový filtr je jedním z nelineárních filtrů, které slouží pro odstranění náhodného šumu z obrazu. Jeho činnost je znázorněna na obr.2.4. V uvažované oblasti jádra,

obvykle 3x3 pixelů, je stanoven medián, který se stává novou hodnotou daného pixelu.



Obr. 2.4: Popis principu mediánového filtru v oblasti jádra 3x3 pixely.

Mediánový filtr odstraňuje skvrnitý a impulzivní šum. V porovnání s Gaussovým vyhlazením je mediánový filtr vhodnější zejména pro aplikace, kde je nutné zachovat ostré hrany. [4]

Obr. 2.5: Nalevo původní šedotónový snímek, napravo snímek vzniklý aplikací mediánového filtru.

2.2 Morfologické operace

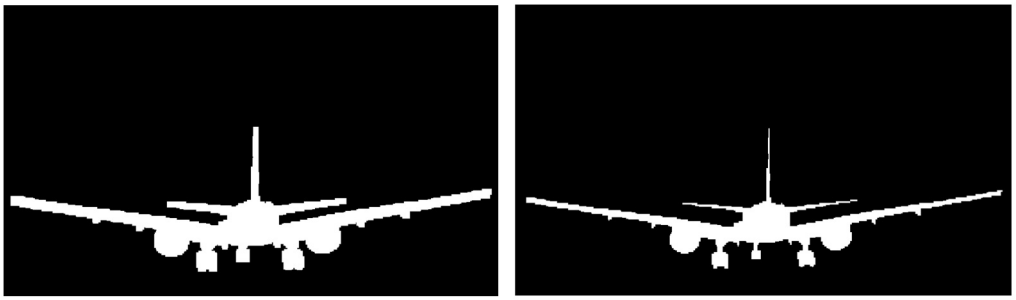
Účelem morfologických operací je především potlačení šumu, odstranění defektů, vyhlazení hran či zvýraznění obrysů objektů. Matematická morfologie pracuje obvykle s binárním obrazem a je též dílčí součástí následujících detekčních metod. [7]

2.2.1 Eroze

Erozi binárního obrazu popisuje rovnice 2.4 , která je definována jako průnik množiny obrazu X s množinou všech posunů masky B

$$X \ominus B = \bigcup_{b \in B}^{n} X_{-b}. \tag{2.4}$$

Eroze obecně zmenšuje objekty, vyhlazuje hrany a odstraňuje malé objekty. [7] Její praktické využití je na obr. 2.6.



Obr. 2.6: Vlevo původní binární snímek, vpravo snímek po erozi s jádrem 5x5 pixelů.

2.2.2 Dilatace

Dilatace binárního obrazu je popsaná rovnicí 2.5 a je definovaná jako sjednocení množiny obrazu X s množinou posunů masky B

$$X \oplus B = \bigcap_{b \in B}^{n} X_{b}. \tag{2.5}$$

Dilatace naopak od eroze zvětšuje objekty, vyhlazuje hrany a vyplňuje malé otvory v objektech. [7] Využívá se pro účely zvýraznění kontur objektu, příklad použití na obr. 2.7.

Obr. 2.7: Vlevo původní binární snímek, vpravo snímek vzniklý dilatací s nastaveným jádrem 5x5 pixelů.

2.3 Základní extrakce příznaků

Jedním z přístupů, jak v obraze vyhledat objekty je základní extrakce příznaků. Tyto metody jsou založeny na hledání rohů a hran, které jsou snadno detekovatelné, jelikož vykazují na rozhraní těchto oblastí vysoký kontrast.

2.3.1 Cannyho hranový detektor

Cannyho hranový detektor je nejčastěji používaná metoda pro detekci hran z důvodu přesné lokalizace s minimem chyb.

Celý proces Cannyho detektoru je širší škála postupů. Nejdříve je aplikováno Gaussovo jádro o velikosti 5x5 pixelů pro odstranění náhodného šumu z obrazu. Pro další výpočet je pak vyhlazený obraz filtrován vhodnou konvoluční maskou. V případě Cannyho detektoru se používá Sobelův operátor v horizontálním

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * I \tag{2.6}$$

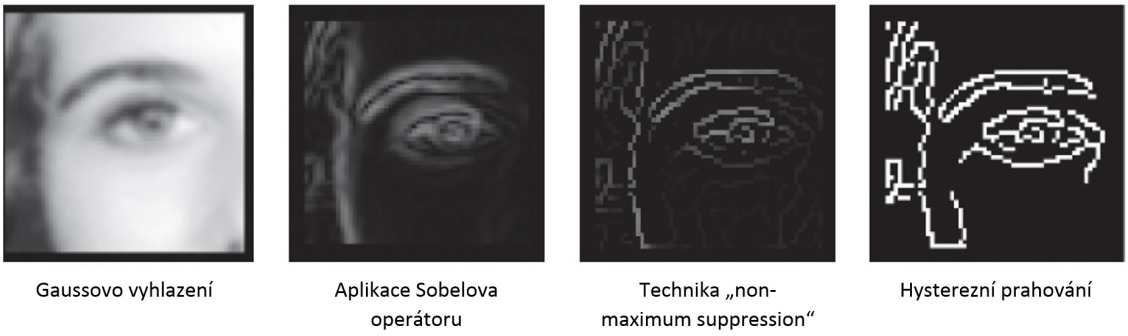
a ve vertikálním směru

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I, \tag{2.7}$$

kde G_x a G_y je konvoluční maska Sobelova operátoru a I je vstupní obraz. Následujícím postupem se určí velikost gradientu hran G a jejich směr θ.

Po konvoluci vstupního obrazu se Sobelovým operátorem však vznikají příliš silné hrany a stále se v obraze vyskuje mnoho chybných pixelů. Z toho důvodu se používá technika *„non-maximum suppression“*. Při jejím použití je pro každý pixel v okolí 3x3 pixelů zkontrolováno, zda se jedná o pixel vykazující lokální maximum vzhledem k jejímu okolí. V případě, že tomu tak není je bod vyřazen.

V dalším kroku jsou eliminovány všechny chybné a nevýznamné hrany. Toho lze docílit pomocí hysterezního prahování s nastaveným minimálním a maximálním prahem. Pokud je posuzovaný pixel mezi těmito dvěma zvolenými prahy, je bod označen jako hrana pouze za podmínky, kdy již sousední bod byl jako hrana uznán. [8][4]



Obr. 2.8: Fáze Cannyho detektoru hran. [4]

2.3.2 Harrisův rohový detektor

Hrany a rohové oblasti vykazují velké změny intenzity do všech směrů. Další z aplikací, které využívají této vlastnosti je Harrisova detekce. Ta funguje na principu hledání rozdílu v intenzitě pro vektor (u, v), který posouváme do všech směrů

$$E(u,v) = \sum w(x,y)[I(x+u, y+v) - I(x,y)]^2], \tag{2.8}$$

kde $w(x, y)$ je aktuální pozice okénkové funkce,
$I(x, y)$ je intenzita v bodě x, y,
$I(x + u, y + v)$ je intenzita v posunutém bodě. [8]

Okénková funkce $w(x, y)$, z rovnice 2.8, je buď obdélníkové nebo Gaussovo okno. Dále lze výraz rozvinout pomocí Taylorovy řady a v maticové podobě vyjádřit změnu gradientu:

$$M = \sum w(x.y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \tag{2.9}$$

kde I_x a I_y jsou derivace intenzity ve směru x a y.

V následující rovnici lze díky hodnotě R určit, zda se v daném okně nachází hrana či roh

$$R = det(M) - k[trace(M)]^2, \tag{2.10}$$

kde

$$det(M) = \lambda_1 \lambda_2 \tag{2.11}$$

$$trace(M) = \lambda_1 + \lambda_2, \tag{2.12}$$

kde λ_1 a λ_2 jsou získány řešením předešlé rovnice.

Platí, že na základě hodnot λ_1 a λ_2 lze určit, zda se v konkrétní oblasti vyskytuje hrana či roh. Pomocí parametru R lze odvodit o jaký typ bodu se jedná:

1. Jestliže $R \approx 0$, stejně jako λ_1, $\lambda_2 \approx 0$ pak se v oblasti nenachází roh ani hrana.
2. Jestliže je $R >> 0$, což nastane jestliže $\lambda_1 >> \lambda_2$ (nebo naopak) je v oblasti detekována hrana.
3. Jestliže $R << 0$. Tato situace nastane pouze tehdy jestliže λ_1,$\lambda_2 >> 0$, pak je v oblasti detekovaná hrana. [8]

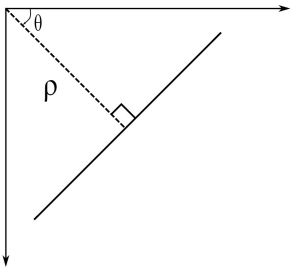
2.3.3 Houghova transformace

Houghova transformace je jednou z technik, která umožňuje detekovat jakýkoliv tvar objektu, který lze matematicky vyjádřit. Tato metoda detekuje i tvary, které jsou přerušované či zkreslené. Houghova transformace je primárně určena pro detekci přímek v obraze, proto ji lze uplatnit například pro detekci horizontu. [27]

Každou přímku lze parametricky vyjádřit normálovou rovnicí přímky

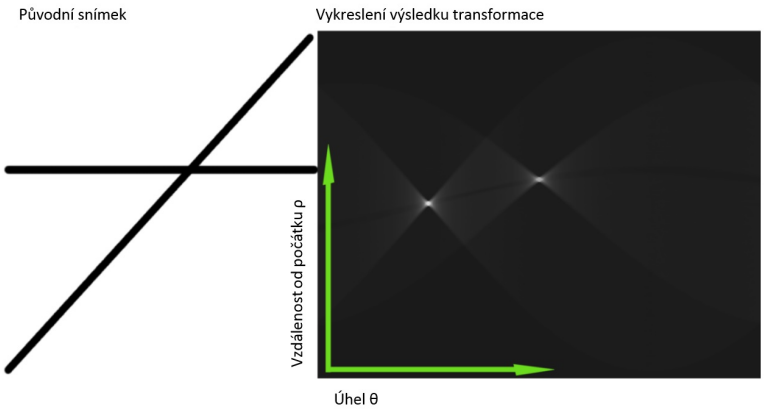
$$\rho = x.cos(\theta) + y.sin(\theta), \tag{2.13}$$

kde ρ je vzdálenost od kolmice přímky k počátku souřadnice a θ je úhel, který svírá osa x s kolmicí dané přímky. [27]



Obr. 2.9: Znázornění odvození normálové rovnice přímky. Inspirováno literaturou [27].

Přímku tak lze vyjádřit dvěma neznámými, a to ρ a θ. Dále je vytvořen tzv. akumulátor, který je iniciován nulovými parametry ρ a θ. Uvažujeme-li snímek, ve kterém jsou body x_i a y_i dosazeny do normálnové rovnice přímky, pak lze nalézt neznámé ρ a θ. Tato transformace bodů na přímku se nazývá právě jako Houghova transformace přímky a je implementována kvantizací parametrů ρ a θ do jednotlivých buněk akumulátoru, které odpovídají dané přímce a jenž se s každou další nalezenou přímkou akumulují. Výsledkem jsou extrémy nalezené v akumulátoru, které vykazují to, že jde o nalezenou přímku. [28]



Obr. 2.10: Akumulátor Houghovy transformace.

3 DETEKCE POHYBU

Detekcí pohybu se zkoumá přítomnost jakékoli změny v daném obraze. Oproti detekcím objektů se zde předem nestanovují konkrétní vlastnosti předmětu. Detekovaný pohyb lze ale přesto ovlivnit různými parametry, například prahem detekce, velikostí detekčního okna anebo stanovením rychlosti pro detekci zkoumaného pohybu.

Pohyb je obvykle některá z kombinací čtyř elementárních pohybů:

1. Translační pohyb v konstantní vzdálenosti od pozorovatele.
2. Translační pohyb v ose kolmé na pozorovatele.
3. Rotační pohyb v konstantní vzdálenosti od pozorovatele.
4. Rotační pohyb v ose kolmé na pozorovatele. [1]

3.1 Diferenční metoda

Princip diferenční metody je založen na jednoduchém odečítání dvou snímků (obr.3.1), které byly získány v různých časových okamžicích. Nezbytným předpokladem pro správnou funkci metody je stacionární pozice kamery, neměnné osvětlení a vysoká frekvence snímků. Výsledný diferenční obraz je dále silně závislý na kontrastu mezi sledovaným objektem a okolím. To znamená, že předmět není možné detekovat pokud by splýval s pozadím. [1]

Obr. 3.1: Dva vstupní po sobě jdoucí snímky pro aplikaci diferenční metody. Vlevo uvažovaný snímek $f_1(x, y)$, vpravo $f_2(x, y)$.

Metoda spočívá v porovnáním dvou po sobě jdoucích snímků, které jsou předtím převedeny do stupní šedi. V dalším kroku se stanoví binární rozdílový obraz $d(x, y)$:

$$d(x,y) = 0 \quad pokud \quad |f_1(x,y) - f_2(x,y)| \geq \varepsilon, \tag{3.1}$$

$$d(x,y) = 1 \quad pokud \quad |f_1(x,y) - f_2(x,y)| < \varepsilon, \tag{3.2}$$

kde ε je práh pro detekci. [1]

Binární rozdílový snímek, znázorněný v pravé části na obr. 3.2, vymezuje oblasti s detekovaným pohybem. Pixely, které vykazují pohyb mají binární hodnotu 1. Pro lepší odlišení oblastí s pohybem je zde zvolen práh ε. Ten zaručí, že algoritmem projdou pouze dostatečně výrazné změny v obraze.

Obr. 3.2: Nalevo diferenční obraz vzniklý rozdílem snímků f_1 a f_2,vpravo binární rozdílový snímek vzniklý prahováním.

Nastavením prahu pro detekci nicméně nelze zabránit chybám. Jestliže uvažujeme dva po sobě jdoucí body, označené jako f_1 a f_2, je také nutné počítat s tím, že může nastat jeden či více z případů chybné detekce. A to z důvodů, že

(a) f_1 je pixel pohybujícího se objektu a f_2 je pixel statického okolí (a opačně).
(b) f_1 je pixel pohybujícího se objektu a f_2 je pixel jiného pohybujícího se objektu.
(c) f_1 je pixel pohybujícího se objektu a f_2 je pixel jiné části stejného objektu. [1]

Výše zmíněné systémové chyby se snažíme co nejvíce potlačit. Ne vždy je ale zvolený práh řešením. Některé chyby nelze vyloučit nikdy, zejména kdy je objekt větších rozměrů. Z tohoto důvodu se používá metoda odečítání pozadí, která využívá porovnání aktuálního snímku a pozadí scény.

3.2 Optický tok

Metoda založená na optickém toku popisuje vektorem (u, v) zvolený pixel v obraze. Tento vektor nese pro každý konkrétní bod informaci o jeho směru a rychlosti. [5] Výpočet optického toku probíhá nejčastěji pro významné body nebo pro rovnoměrně rozprostřené body v obraze. Příkladem je obr. 3.3, na kterém se nachází rovnoměrně rozprostřená mřížka vektorů. Každý vektor znázorňuje pro určitý pixel směr pohybu a jeho rychlost. Stejně jako diferenční metoda nepracuje explicitně s barevnými informacemi. [1].

Obr. 3.3: Snímek s vypočtenými vektory optického toku. [9]

Metoda pracuje za předpokladu, kdy vybraný pixel $f(x, y, t)$ v počátečním okamžiku je posunut o vzdálenost dx, dy za čas dt a jeho hodnota se neměnní:

$$f(x, y, t) = f(x + dx, y + dy, t + dt). \tag{3.3}$$

Pixel $f(x, y, t)$, vyjádřený v předešlé rovnici, je tak v čase neměnný a má stále stejnou intenzitu. [8]

Rovnice optického toku je posléze získána aproximací pravé strany Taylorovou řadou a eliminací členů vyšších řádů:

$$f_x u + f_y v + f_t = 0. \tag{3.4}$$

Výsledný vektor (u, v) zůstane neznámý, jelikož není možné řešit jednu rovnici o dvou neznámých. Proto se pro určení vektoru optického toku nejčastěji používá výpočetní metoda *„Lucas-Kanade“*. [8]

Metoda *„Lucas-Kanade“* pracuje za předpokladu, že okolní pixely vykonávají stejný pohyb. Přitom je využito okolí 3x3 bodů. Za pomoci 9 rovnic o dvou neznámych je tak možné vyřešit rovnici optického toku.

Vlevo na obr. 3.4 je zobrazen tzv. hustý optický tok, kde vektor optického toku je spočten pro všechny body obrazu, v pravé části výsledný detekovaný objekt.

Obr. 3.4: Ukázka z praktické realizace. Vlevo obraz hustoty optického toku, vpravo výsledný detekovaný objekt.

3.3 Metoda subtrakce pozadí

Metoda substrakce pozadí (z angl. *„Background Subtraction“*) je jednou z technik pro detekci pohybu. Na rozdíl od metody odečítání pozadí, zde není předem definován snímek pozadí. Nahrazuje tak metodu odečítání pozadí na základě diferenční metody, se kterou jsou spojeny problémy zejména v případě, jestliže není snímek pozadí aktuální. Takový algoritmus nedokáže rozlišit skutečné pozadí od okolí. Na rozdíl od toho, metoda substrakce pozadí pracuje s reálnými snímky, kterým je přiřazena různá váha a obraz pozadí je vytvářen za chodu.

Pokud se na snímku čas od času nacházejí drobné změny, například mraky či stíny, pak je vhodné snímek pozadí průběžně aktualizovat. Dochází tak k průběžnému průměrování snímku

$$f_{BG} = (1 - \alpha) \cdot f_{BG} + \alpha \cdot f_{AF}, \tag{3.5}$$

kde f_{BG} je snímek pozadí, f_{AF} je aktuální snímek a α je váha aktuálního snímku, který tvoří část celkového snímku pozadí. Nejběžněji se α volí mezi 0,05 až 0,1. Čím vyšší hodnota α, tím více je dána váha na aktuální snímek, a proto nemusí být detekovány drobnější změny v obraze. [29]

Dále již metoda navazuje na princip diferenční metody, kde výsledný obraz f_{AD} je dán absolutní hodnotou rozdílu aktuálního snímku f_{AF} od váhovaného snímku pozadí f_{BG}

$$f_{AD} = |f_{AF} - f_{BG}|. \tag{3.6}$$

Na obr. 3.5 se vlevo nachází průběžný rozdílový snímek f_{AD} a vpravo původní obraz.

Obr. 3.5: Nalevo diferenční obraz vzniklý rozdílem snímků f_1 a f_2, napravo původní snímek.

4 DETEKCE A ROZPOZNÁVÁNÍ OBJEKTŮ

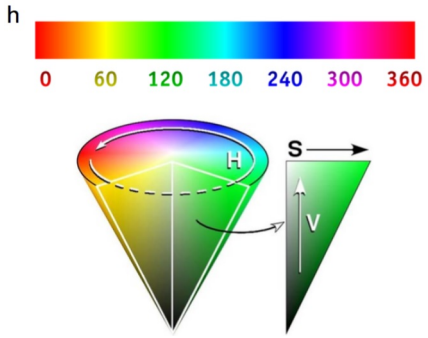
V této kapitole je popsána detekce objektů, která přímo nezávisí na jeho pohybu, ani rychlosti. Rozdílem oproti pohybovým detekcím je zejména v možnosti dynamicky pohybovat kamerou. Absence pohybových parametrů je nahrazena využitím jiných vlastností, které jsou klíčové pro daný objekt. Mezi takové vlastnosti patří významné body, oblasti, tvar a barva. Do této problematiky lze zahrnout i rozpoznávání objektů, které se děje na základě předem poskytnutého snímku, popř. více snímků. Takový algoritmus dokáže rozlišovat objekty na základě předem naučeného klasifikátoru.

4.1 Detekce podle barvy

Tato metoda detekuje objekt na základě předem vybraného úseku barevného spektra. Za účelem vytvoření přesného detektoru je nutné zvolit vhodný formát popisu barevného prostoru. A to z důvodu, že následnou detekci znesnadňují rozdílné intenzity jasu, které způsobují například stíny a odrazy. Právě pro takový účel detekce volíme barevný prostor HSV, jehož hlavní výhodou je oddělení barevné složky od jasové. [10] Převod hodnot z barevného prostoru RGB do HSV lze popsat následovně:

$$H = 60 \times \begin{cases} 0 + (\frac{G-B}{max-min}) & pokud\ max = R, \\ 2 + (\frac{B-R}{max-min}) & pokud\ max = G, \\ 4 + (\frac{R-G}{max-min}) & pokud\ max = B. \text{[11]} \end{cases} \tag{4.1}$$

Na obrázku 4.1 je znázorněn kuželový grafický model HSV. Barva je stanovena proměnnou H, která označuje barevný tón. Parametr S představuje sytost barvy a proměnná V množství bílého světla.



Obr. 4.1: Kuželový model barevného prostoru HSV. [11]

4.2 Korelační metoda

Korelační metoda je jednou z detekčních technik, která vyhledává ve vstupním snímku všechny oblasti, které se shodují se zpravidla menší předlohou.

Pro správnou funkci jsou zapotřebí dva snímky, a to vstupní obraz $f(x, y)$ a předloha $t(x, y)$, tyto snímky se nacházejí na obr. 4.2. Nalevo v horní části je předloha pro detekci. Obraz určený pro korelaci se nachází ve spodní části obrázku. Vpravo je na obrazy aplikován Cannyho hranový detektor pro získání lepších parametrů detekce.

Obr. 4.2: Vlevo nahoře šablona $t(x, y)$ určená pro korelaci se vstupním obrazem $f(x, y)$ dole vlevo. Na pravé straně jsou snímky po použití Cannyho detektoru.

Dále proběhne výpočet korelace z normovaných hodnot jasu obrazů $f(x, y)$ a $t(x, y)$

$$n(x, y) = \sum_{x,y \in R} \hat{f}(x, y)\hat{t}(x, y), \tag{4.2}$$

kde $n(x, y)$ je normovaná korelace (vlevo na obr. 4.3), která nabývá hodnot v rozmezí 0 až 1. Posléze je stanoven práh pro detekci, který bude lokalizovat všechny oblasti s dostatečnou shodou předlohy se vstupním obrazem. [8][12]

Obr. 4.3: Vlevo obraz normované korelace. Místo s největší shodou je označeno nejjasnějším bodem. V tomto místě se nachází levý horní roh snímku. Vpravo detekovaný objekt označený obdélníkem podle šířky a výšky předlohy.

Z důvodu jednoduchosti svého principu vykazuje i řadu problémů. Mezi největší patří závislost na měřítku šablony a na změny intenzit jasu. Dále není invariantní vůči rotaci a afinní transformaci. To znamená, že i při malém natočení nastává problém s detekcí. [13]

Korelační metoda pracuje velmi efektivně pouze za předpokladu, že šablona a vstupní obraz jsou po celou dobu stabilní. Toho lze docílit například pomocí Cannyho hranového detektoru, který dobře definuje obrazy s vyznačenými hrany [14], vpravo na obr. 4.2.

4.3 Detekce podle zájmových bodů

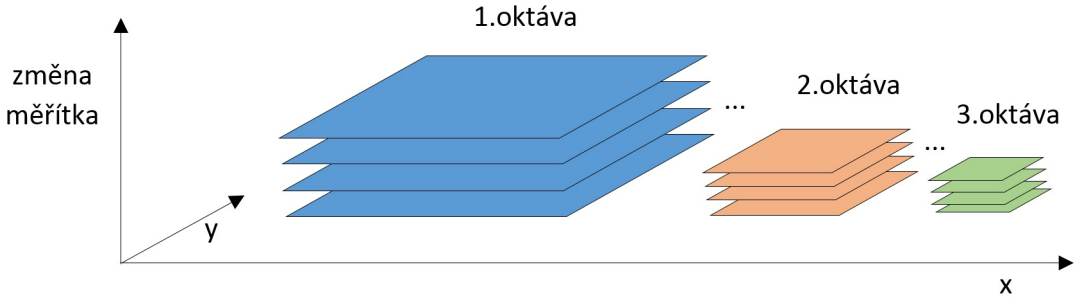
Detekce zájmových bodů zahrnuje větší počet metod. Základní princip spočívá v analýze specifických bodů a jejich následném zpracováním. Vhodně vybraný bod by měl

(a) být zřetelný a dobře matematicky definovatelný,
(b) být významný z lokálního hlediska,
(c) být opakovatelný,
(d) mít relevantní informační obsah. [9]

4.3.1 Metoda SIFT

Pod zkratkou SIFT (z angl. *Scale-Invariant Feature Transform*) se označuje metoda používaná za účelem detekce objektů na základě lokálních příznaků. Hlavní přednostní je především její nezávislost na měřítku.

V prvním kroku jsou lokalizovány významné body pomocí detekce extrémů uvnitř měřítkově nezávislého prostoru, který je zobrazen na obr. 4.4. V této fázi je sestavena vícevrstvá reprezentace obrazu, kde první vrstva odpovídá původnímu obrazu a každá další je tvořena obrazem o větším měřítku. Dále jsou sestaveny oktávy, kde každá další reprezentuje obraz o poloviční velikosti.



Obr. 4.4: Měřítkový prostor scale-space složený z jednotlivých oktáv. Inspirace z literatury [16]

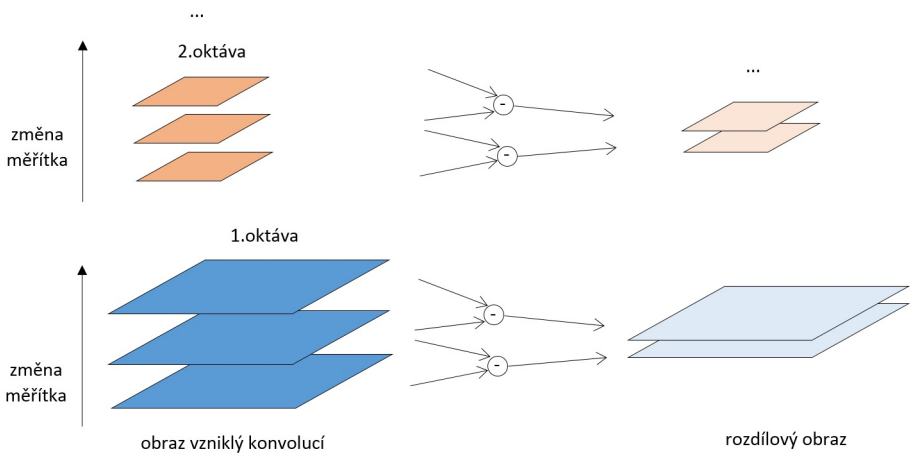
Po sestavení měřítkově nezávislé reprezentace je využita Gaussova pyramida, která vznikne konvolucí Gaussova jádra $G(x, y, \sigma)$ se vstupním obrazem $I(x, y)$:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}, \tag{4.3}$$

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{4.4}$$

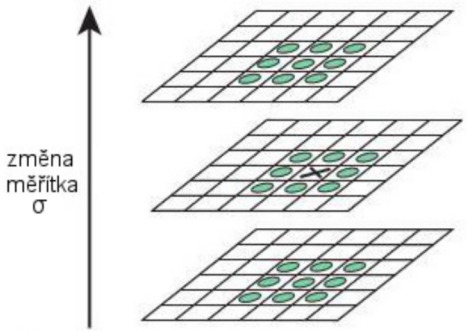
kde σ je šířka Gaussova jádra, které představuje změnu měřítka. [8]

Z konstrukce měřítkového prostoru, ilustrovaného na obrázku 4.4, vzniknou pomocí rozdílů dvou sousedních obrazů Gaussovy pyramidy $L(x, y, \sigma)$ z měřítkově nezávislé reprezentace diferenční obrazy $D(x, y, \sigma)$, znázorněné vpravo na obr. 4.5.



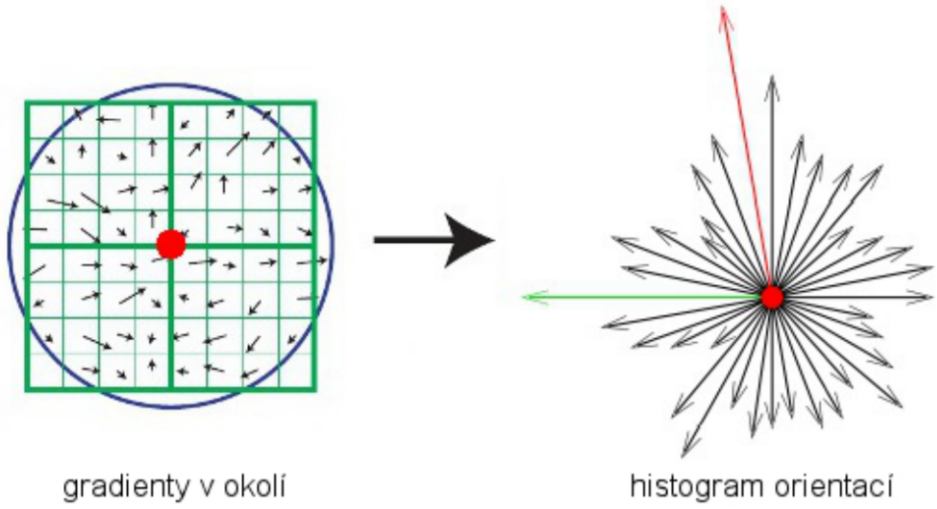
Obr. 4.5: Sestavení rozdílového obrazu $D(x, y, \sigma)$ z obrazu vzniklý konvolucí $L(x, y, \sigma)$. Inspirace z literatury [16]

V rozdílových obrazech jsou následně vyhledávány lokální extrémy napřič 26 body v okolí. Tyto body zahrnují osm sousedních bodů ve stejném měřítku, devět bodů v následujícím měřítku a stejný počet v předešlém měřítku.



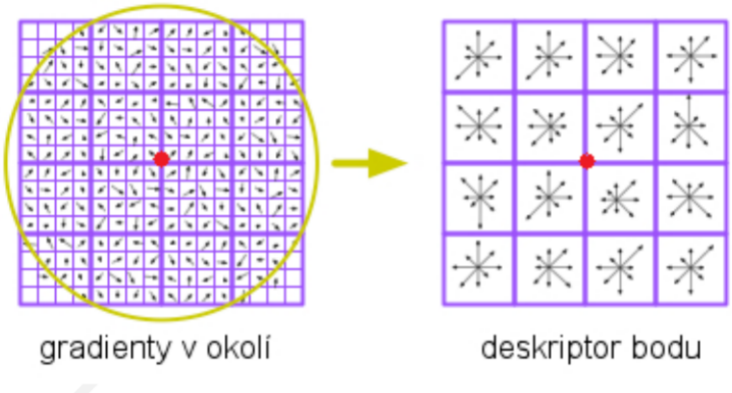
Obr. 4.6: Detekce lokálních extrémů. Zvolený pixel X je porovnáván se svým okolím. [8]

Z vypočtených hodnot gradientů v okolí významného bodu, vlevo na obr. 4.7, je dále zkonstruován histogram orientací. Tento histogram dohromady čítá 36 binů, což jsou jednotlivé intervaly histogramu a pokrývají tak veškerý rozsah rotace. Dominantní orientace je přitom určena globálním maximem, znázorněným červenou šipkou na obr.4.7



Obr. 4.7: Vlevo vypočtené gradienty v okolí významného bodu vážené Gaussovým oknem, vpravo sestavený histogram orientací.[16]

Po přiřazení dominantní orientace významným bodům vstupuje do procesu tzv. deskriptor, zobrazen v levé části obr.4.8. Jeho úkolem je popsat okolí významných bodů nezávisle na geometrických a jasových transformacích. Výsledný deskriptor na obr. 4.8, je označován jako 128-binový. [3][8]



Obr. 4.8: Sestavený deskriptor metody SIFT. [16]

4.3.2 Metoda SURF

Metoda SURF (z angl. *Speeded-Up Robust Features*) vychází z koncepce metody SIFT. Důvodem použití je záměr vytvořit výpočetně rychlejší a stabilnější algoritmus. Nevýhodou je, že tento algoritmus je stejně jako u metody SIFT patentově

chráněný.

Při aproximaci rozdílů Gaussovou funkcí má metoda SIFT příliš velké odezvy v okolí hran, kde dochází k chybné detekci. Tyto chybné body jsou pak odstraněny pomocí Hessianovy matice. Naopak metoda SURF kombinuje tyto dva kroky přímým výpočtem pomocí determinantu Hesissianovy matice. [16]

Integrální obraz

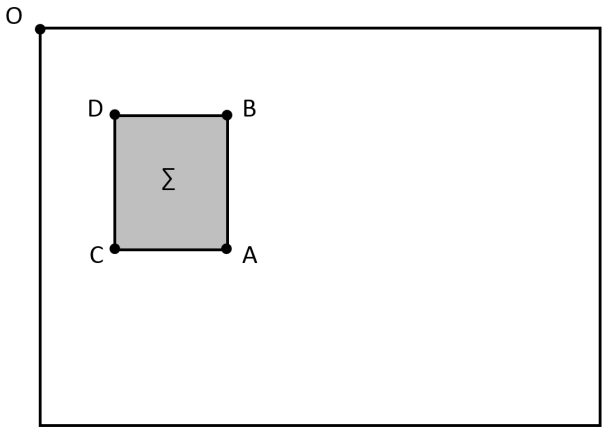
Pomocí integrálního obrazu je možné získat údaje o intenzitě v dané oblasti, a to s pouhou znalosti krajních bodů. Integrální obraz umožňuje rychlý součet hodnot uvnitř libovolné obdélníkové oblasti vstupního obrazu. [16]

$$I_{\sum}(x,y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i,j), \tag{4.5}$$

kde $I(i,j)$ je vstupní obraz a $I(x,y)$ je integrální obraz. Na obr. 4.9 je znázorněna oblast, která je uvažována pro součet hodnot v obdélníkovém regionu. Výpočet integrálního obrazu lze snadno vyjádřit jako

$$\sum = A - B - C + D, \tag{4.6}$$

kde $\sum$ je hledaný součet a body A, B, C, D jsou hodnoty integrálního obrazu v daných souřadnicích.



Obr. 4.9: Výpočet součtu hodnot v regionu $\sum$ pomocí integrálního obrazu

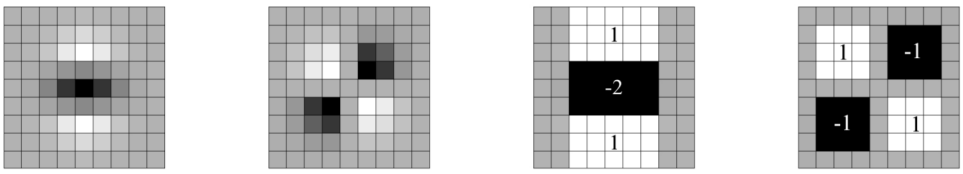
Hessianova matice

Hessianova matice bodu (x, y) je v daném měřítku definovaná jako:

$$H(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{yx}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix}, \tag{4.7}$$

kde $L_{xx}(x, y, \sigma)$ je hodnota druhé parciální derivace podle x a σ je šířka Gaussova jádra.

Pro výpočet prvků Hessianovy matice se využívají obdélníkové funkce. Jedná se o takové funkce, které aproximují konvoluci obrazu s filtry na obr. 4.10. Tato implementace umožňuje rychlejší výpočet. [16]



Obr. 4.10: Nalevo druhá derivace Gaussovy funkce podle y a x, y. Napravo aproximace pro druhou derivaci Gaussovy funkce podle y a x, y. [15]

Sestavení „scale-space“

Metoda SURF využívá princip konstrukce *„scale-space“* složenou z jednotlivých obrazů vzniklých filtrací. Složitost výpočtů je konstantní z důvodů integrálního obrazu. Nutnost generování kaskádovitého „scale-space“ zde odpadá, čímž je dosažena vyšší rychlost v porovnání s metodou SIFT. [16]

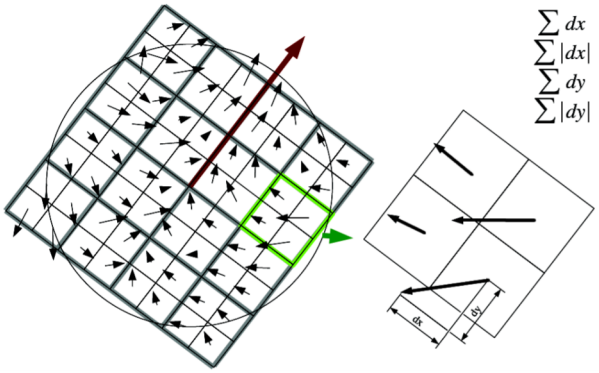
Detekce maxim a přiřazení orientace jednotlivým bodům

Detekce maxim uvnitř *„scale-space“* probíhá stejným způsobem jako u metody SIFT. Body jsou srovnány s 26 v okolí. Pokud má nalezený bod ve svém okolí nejvyšší hodnotu, pak je považován za významný.

Metoda SURF přiřazuje každému významnému bodu jeho dominantní orientaci tak, aby výsledný deskriptor byl invariantní vůči rotaci. Na rozdíl od metody SIFT zde není využíván histogram orientací, ale zkoumá se odezva na Haarovu vlnku v kruhovém okolí daného bodu. Výsledek konvoluce obrazu s Haarovou vlnkou je dále vážen koeficienty Gaussova kruhového okna se středem ve zkoumaném bodě. [16]

Sestavení deskriptoru

Sestavený deskriptor (vlevo na obr. 4.11) se skládá ze získaných vektorů pro 16 podoblastí a popisuje ho 64 binů, to znamená 16 podoblastí vynásobených 4-prvkovým vektorem (vpravo na obr. 4.11). [15]



Obr. 4.11: Vlevo výsledný deskriptor metody „SURF“, vpravo čtvercová podoblast 4x4 pixely s odezvy na Haarovu vlnku[15]

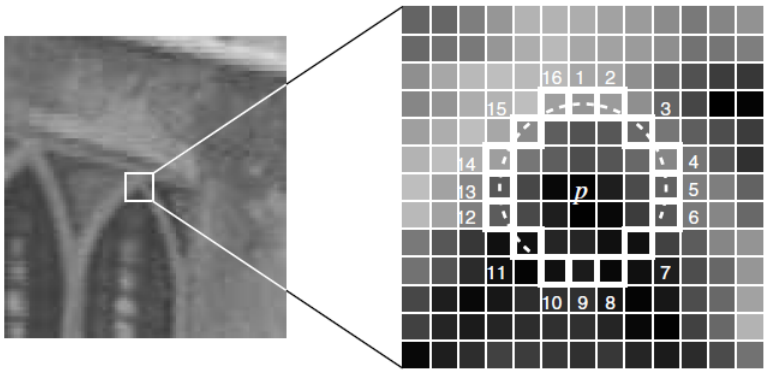
4.3.3 Metoda ORB

Jednou z alternativ k patentově chráněným metodám SIFT a SURF je algoritmus ORB (z angl. *Oriented FAST and Rotated BRIEF*). Tato metoda vznikla na základě FAST detektoru a BRIEF deskriptoru.

Metoda ORB přináší řadu výhod. Přední výhodou je menší náročnost na výpočetní výkon a s tím i rychlejší zpracování. Proto je možné ji použít i v aplikacích pracujících v reálném čase. [8]

Jak už bylo zmíněno, metoda ORB je založena na detektoru významných bodů FAST (z angl. *Features from accelerated segment test*), Jedná se o segmentový test, kde uvažujeme kruhovou oblast šestnácti pixelů kolem hledaného rohu p (na obr.

4.12). Uvažovaný bod p je považován za rohový segment. Rozhodující pro tento test je podmínka nalezení 12 kontinuálních pixelů v kruhové oblasti, které jsou světlejší než $f(p) + t$, kde $f(p)$ je intenzita v bodě p a t je zvolený práh. a nebo jestliže jsou všechny body tmavší než $f(p)$. [17]



Obr. 4.12: Lokalizace rohových pixelů pomocí segmentového testu.[18]

Z důvodu absence rotační invariance v původní metodě FAST, se posléze přidává a je definovaná jako

$$\theta = arctg2\Big(\sum_{x,y} x \cdot f(x,y), \sum_{x,y} y \cdot f(x,y)\Big). \tag{4.8}$$

Deskriptor BRIEF (z angl. *Binary Robust Independent Elementary Features*) využívá jednoduchého binárního testu mezi dvěma pixely

$$\tau(p; x, y) = \begin{cases} 1 & p(x) < p(y) \\ 0 & p(x) > p(y) \end{cases}, \tag{4.9}$$

kde $p(x)$ je intenzita bodu p po aplikování vyhlazovacího jádra v bodě x. Tento binární test provede pro n_d lokačních párů, abychom dostali n_d-dimenzionální binární řetězec. Zvolené n_d může být 128, 256 nebo 512 v závislosti na požadované přesnosti a rychlosti zpracování. [17]

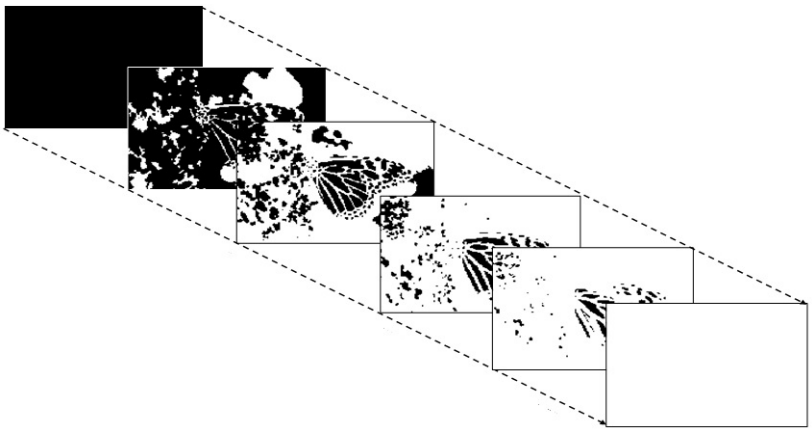
4.4 Detekce významných oblastí

Podobným způsobem jako lze pro detekci použít významné body, můžeme využít vlastností zájmových oblastí. Aby dané oblasti byly detekovatelné, musí splňovat určitá kritéria. Důležitým předpokladem je, aby takové oblasti vykazovaly stabilitu, nezávislost změny osvětlení a natočení obrazu.

4.4.1 Metoda MSER

Metoda MSER (z angl. *Maximally stable extremal regions*) umožňuje detekci na základě lokalizace maximálně stabilních oblastí.

Princip metody je znázorněn na obr. 4.13 a spočívá v prahování šedotónového obrazu přes širokou škálu prahů. Tím lze získat jednotlivé binární snímky, kde pixely překračující práh mají hodnotu 1. Jestliže v prvním kroku zvolíme minimální práh, dostaneme obraz vyjádřený pouze binární hodnotou 1. To znamená, že bude celý bílý. Dále pokud se práh bude zvyšovat, dojde k postupnému objevování černých skvrn, jenž indikují lokální minima jasové funkce obrazu. Veškeré uzavřené oblasti, které se během všech prahovacích kroků objeví, jsou označeny jako maximální regiony.



Obr. 4.13: Ilustrace znázorňující prahování obrazu u metody MSER. [19]

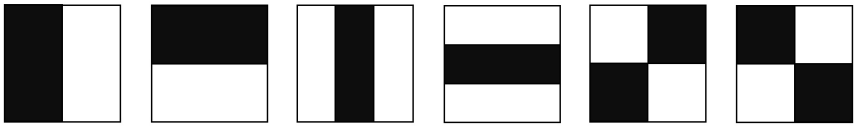
Jestliže taková oblast vykazuje stabilitu v širokém rozsahu prahů, pak je i invariantní vůči jasovým transformacích a je označena za maximálně stabilní oblast. [16]

4.5 Detektor „Viola-Jones“

Využití detektoru „Viola-Jones“ je výhodné vzhledem k výpočetní rychlosti. Tento způsob detekce je velmi spolehlivý a zároveň umožňuje využití i v aplikacích pracujících v reálném čase. [20]

Haarovy příznaky

Algoritmus pro natrénovaní klasifikátoru potřebuje mnoho pozitivních a negativních snímků. Za účelem získání příznaků z těchto snímků jsou použity odezvy na Haarovy filtry, znázorněné na obr. 4.14. Jejich funkce je obdobná jako u použití konvolučního jádra. Hodnota příznaku je získána rozdílem sum jasových hodnot pod světlou oblastí od sumy jasových hodnot pod tmavší oblasti. [8]



Obr. 4.14: Základní typy Haarových příznaků.

Integrální obraz

Převod snímku na integrální obraz zajistí rychlý a efektivní výpočet zkoumaných obdélníkových Haarových příznaků. Integrální obraz v bodě x, y je suma pixelů ohraničené oblasti těmito body zdola a zprava

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'), \tag{4.10}$$

kde $i(x', y')$ je hodnota pixelu původního snímku. $ii(x, y)$ je danému pixelu snímku odpovídající integrální obraz. Hodnota čtvercové oblasti snímku $ABCD$ lze vypočítat ze čtyř hodnot integrálního obrazu, stejně jako v případě předešlé metody SIFT. [20]

Algoritmus AdaBoost

Je-li k dispozici sada příznaků a soubor pozitivních a negativních snímků, jsme schopni určit funkci klasifikátoru. Pozitivní jsou takové snímky, na kterých se nachází pouze zkoumaný objekt. Jako negativní snímky mohou být naopak použiti jakékoliv snímky splňující jedinou podmínkou a to, že se daný objekt na nich nenachází a to ani částečně. Algoritmus AdaBooost využívá výběru malého počtu příznaků k natrénovaní klasifikátoru. Jednoduchý AdaBoost klasifikátor je váhovaná suma mnoha slabých klasifikátorů, kde každý ze slabých klasifikátorů je prahován přes jeden obdélníkový Haarův příznak. [20]

Slabý klasifikátor je definován jako:

$$h(x, f, o, \theta) = \begin{cases} 1 & pf(x) < p\theta \\ 0 & jinak, \end{cases} \tag{4.11}$$

kde f označuje funkční hodnotu slabého klasifikátoru, θ je práh, p je polarita nerovnosti.

Proces učení algoritmu AdaBoost je popsán následovně:

1. Na trénovacích snímcích $(x_1, y_1)...(x_n, y_n)$ jsou označeny $y_i = 0$ negativní vzorky, resp. pozitivní.
2. Inicializuje se klasifikátor počtem $t = 0$ s počáteční váhou vzorku $w_i = \frac{1}{2m}$ pro $y_i = 0$, resp. $w_i = \frac{1}{2l}$ pro $y_i = 1$, kde hodnoty m a l značí počet negativních a pozitivních snímků.
3. Pokud je počet vyřazených negativních vzorků menší než 50%:
   (a) Inkrementuje se $t = t + 1$.
   (b) Stanoví se nová hodnota váhy $w_i = \frac{w_i}{\sum_j w_j}$.
   (c) Vybere se nejlepší slabý klasifikátor s ohledem na váženou chybu $\epsilon_t = \min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$.
   (d) Definuje se $h_t(x) = h(x, f_t, p_t, \theta_t)$, kde f_t, p_t a θ_t jsou minima funkce ϵ_t.
   (e) Znovu vypočteme váhu $w_i = w_i \beta_t^{1-e_i}$, kde $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ a $e_i = 0$, pokud je x_i správně určen, resp. $e_i = 1$ v opačném případě.

(f) Silný klasifikátor je definován:

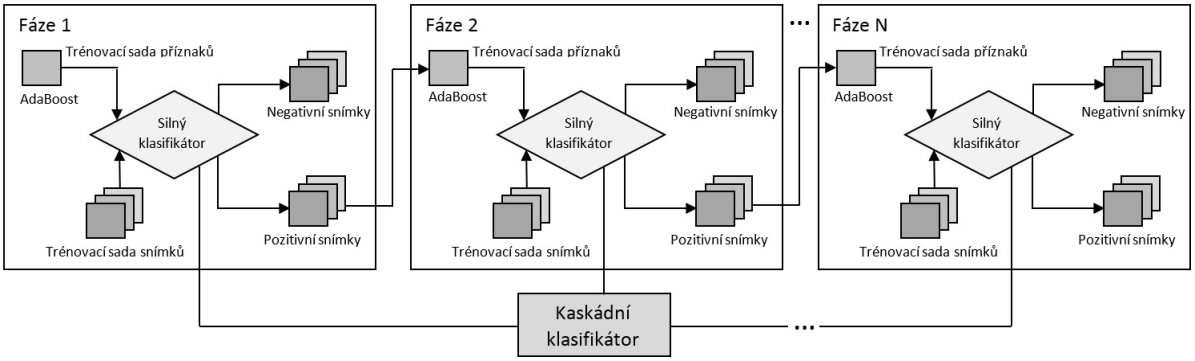
$$H(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \gamma_t \\ 0 & jinak, \end{cases} \tag{4.12}$$

kde α_t je váha slabého klasifikátoru $\alpha_t = \log \frac{1}{\beta_t}$ a γ_t je vybrán tak, aby všechny pozitivní vzorky byli správně klasifikovány.

(g) Výpočet nových vzorků z předešlých, které prošly předešlou kaskádou klasifikátorů. [20]

Kaskádní klasifikátory

Kaskádní klasifikátor, znázorněný na obr. 4.15 je složen z jednotlivých stupňů, které obsahují slabé klasifikátory. Práce každého stupně spočítá v rozřazení daného snímku mezi negativní vzorky, které jsou ihned vyřazeny, nebo možné pozitivní. Negativní vzorky již dále nepostupují. Snímky, které jsou klasifikovány jako možné pozitivní dále postupují do další fáze v kaskádě. Výhodou je, že následující stupně vykazují vyšší šanci na úspěch pozitivní detekce. [20]



Obr. 4.15: Kaskádní klasifikátor vzniklý z dílčích slabých klasifikátorů. Inspirace z literatury [20]

5 TESTOVÁNÍ ALGORITMŮ

V této kapitole jsou testovány jednotlivé metody. Posuzovány jsou z hlediska preciznosti detekce, možností využití a rychlosti zpracování. V této části jsou na testování zvoleny takové videosnímky, aby se na nich vyskytovali ptáci, letadla či drony. Všechny algoritmy v rámci praktické části byly programovány v jazyce Python na Raspberry Pi 3B+ s využitím knihovní funkce OpenCV.

5.1 Algoritmy založené na diferenční metodě

Do této kategorie lze zařadit metodu odečítání pozadí a rozdílovou metodu. U diferenční metody je vypočten rozdíl dvou po sobě jdoucích videosnímků, naopak u metody odečítání pozadí je vyjádřen rozdíl z aktuálního snímku a obrazu pozadí.

Po načtení vstupních dat se obrazy převedou z barevné palety na šedotónový snímek a následně proběhne výpočet rozdílového obrazu za pomocí funkce `cv2.absdiff()` pro oba snímky.

Výpis 5.1: Výpis části kódu pro metodu odečítání pozadí

```
dif = cv2.absdiff(actual_frame, background)                    1
thresh = cv2.threshold(dif, 5, 255, cv2.THRESH_BINARY)[1]      2
thresh = cv2.dilate(thresh, None, iterations=2)                3
```

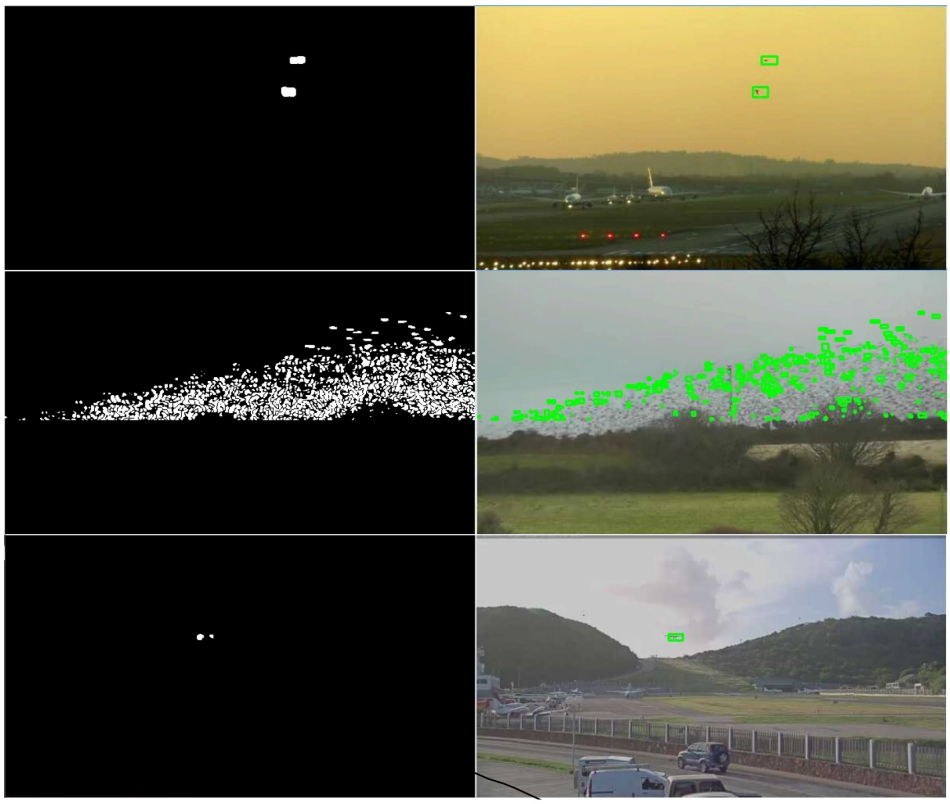
Dále je rozdílový obraz prahován funkcí `cv2.threshold()` s prahem v rozsahu od 0 do 255. Oblasti s pohybem jsou v binární rozdílovém snímku označeny bílou barvou. Na binární snímek je aplikována dilatace `cv2.dilate()` s jádrem 3 x 3 pixely. Dilatace je provedena ve dvou iteracích, aby dostatečně zvýraznila oblasti s detekovaným pohybem.

Výpis 5.2: Výpis kódu pro lokalizaci oblastí s pohybem a jejich označení

```
(_, cnts, _) = cv2.findContours(thresh.copy(),                 1
 cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)                   2
for c in cnts:                                                 3
  (x,y,w,h) = cv2.boundingRect(c)                              4
  cv2.rectangle(actual_frame,(x,y),(x+w,y+h),(0, 255, 0),2)    5
```

Závěrem jsou vyhledány oblasti, pomocí funkce `cv2.findContour()`, které označují pohyb (na obr. 5.1). Tyto oblasti jsou vyznačeny zeleným obdélníkem.

Snímky s detekovanými objekty lze uložit a pomocí dalších podprogramů zasílat upozornění.



Obr. 5.1: V levé části binární rozdílové snímky vzniklé diferenční metodou, napravo výstupní obrazy s detekovanými objekty

Výpočetní náročnost dílčích procesů pro jeden snímek videa byla spočtena jako průměrná hodnota z každého desátého snímku ve videu, aby byla zaručena objektivnost měření. K tomuto byla využita funkce z modulu `datetime`.

Při porovnání hodnot v tabulce 5.1 lze říct, že rychlost zpracování se zásadně neliší. Předzpracování obrazu je časově nejnáročnější operace, a to z důvodu nutnosti převodu snímků do stupní šedi a jejich vyhlazení Gaussovým jádrem. Poté již proces rozdílového obrazu je pouhé odečtení dvou matic. Další procesy jako vykreslení a uložení snímku jsou již z hlediska časové náročnosti nepodstatné. Celkově lze dosáhnout u těchto metod počtu 10 až 11 snímků za sekundu.

Tab. 5.1: Srovnání výpočetní náročnosti pro diferenční metody.

Diferenční metoda	čas [ms]
Předzpracování snímku	70,831
Výpočet rozdílového obrazu	11,405
Vyznačení oblasti s pohybem	7,739
Celkový čas pro jeden snímek	89,975

Metoda odečítání pozadí	čas [ms]
Předzpracování snímku	75,069
Výpočet rozdílového obrazu	15,064
Vyznačení oblasti s pohybem	9,312
Celkový čas pro jeden snímek	99,445

5.2 Výpočet optického toku

Algoritmy pro výpočet optického toku lze dělit na dvě skupiny, a to na výpočet řídkého a hustého optického toku. [8] Pro tuto práci byl zvolen hustý optický tok, jehož výpočet probíhá pro všechny body v obraze.

Tato metoda používá stejný princip předzpracování obrazu jako u diferenčních metod. Výpočet optického toku pak probíhá za pomocí funkce `cv2.calcOpticalFlowFarneback()`. V parametrech funkce se nastavují dva vstupní 8-bitové šedotónové snímky, dále parametr specifikující velikost pyramidového schéma, počet pyramid, velikost okna a počet iterací.

Výpis 5.3: Výpis kódu pro výpočet optického toku

```
flow = cv2.calcOpticalFlowFarneback(prvs,nextf,None, 0.5, 1, 15, 3,
                                    10, 1.1, 0)
mag, ang = cv2.cartToPolar(flow[...,0], flow[...,1])
hsv[...,0] = ang*180/np.pi/2
hsv[...,2] = cv2.normalize(mag,None,0,255,cv2.NORM_MINMAX)
rgb = cv2.cvtColor(hsv,cv2.COLOR_HSV2BGR)
```

Návratovou hodnotou funkce `cv2.calcOpticalFlowFarneback()` je dvourozměrné pole optického toku (u, v). Teoreticky jde o vyjádření v kartézské soustavě, které říká o kolik se daný bod posunul. Tento výsledek je dále převeden do polárních souřadnic za pomoci `cv2.cartToPolar()`. Tato funkce má návratovou hodnotu v radiánech. Na základě směru a velikosti změny je vybrána hodnota z barevné palety HSV. Parametr *„Hue“* vyjadřuje směr pohybu a *„Value“* hodnotu jasu.

Vpravo dole na obr. 5.2 je obraz hustého optického toku, je na těm patrné, že různé směry jsou odlišeny barvou. Z důvodu nadbytečnosti této informace o směru pohybu je snímek převeden na šedotónový obraz a dále prahován, jako v předchozím případě.

Obr. 5.2: Detekce pomocí hustého optického toku. Vlevo výstupní obraz, vpravo dole obraz hustého optického toku, vpravo nahoře prahovaný obraz překrytý maskou v oblasti letiště.

Pro výpočet hustého optického toku, před voláním funkce `cv2.calcOpticalFlowFarneback()`, lze obraz optimalizovat na vhodné rozlišení, aby byla zajištěna přesnost, ale zároveň také rychlý výpočet. Při výpočtu optického toku lze dosáhnout počtu 7 - 8 snímků za sekundu.

Tab. 5.2: Srovnání výpočetní náročnosti optického toku při různém rozlišení.

Videosnímek 400x225 px	čas [ms]
Předzpracování snímku	35,967
Výpočet optického toku	356,801
Vyznačení oblasti s pohybem	51,445
Celkový čas pro jeden snímek	444,213

Videosnímek 200x112,5 px	čas [ms]
Předzpracování snímku	41,855
Výpočet optického toku	64,354
Vyznačení oblasti s pohybem	20,145
Celkový čas pro jeden snímek	126,354

5.3 Detekce maximálně stabilních extrémních oblastí

Při detekci objektů je možné vycházet z oblastí jenž vykazují extrém jasové funkce. Typicky jsou objekty na obloze vůči pozadí kontrastní a splňují podmínku stability přes širokou škálu prahů.

Maximálně stabilní extrémní oblasti jsou vyhledány za pomocí `mser.detectRegions()` ze vstupního snímku. Návratovou hodnotou jsou body ohraničující tuto oblast. V dalším kroku je provedeno ohraničení pomocí konturové aproximace `cv2.convexHull()`.

Výpis 5.4: Výpis kódu pro určení maximálně stabilních extrémních oblastí

```
regions = mser.detectRegions(gray, None)
hulls = [cv2.convexHull(p.reshape(-1, 1, 2)) for p in regions]
```

Za účelem vykreslení detekovaných oblastí je vytvořena maska o velikosti vstupního snímku. Ohraničené oblasti jsou na této masce vyznačeny bíle. Pro názornost lze masku logicky násobit se vstupním snímkem.

Výpis 5.5: Výpis kódu pro vykreslení maximálně stabilních extrémních oblastí

```
mask = np.zeros((img.shape[0], img.shape[1], 1), dtype=np.uint8)
for contour in hulls:
  cv2.drawContours(mask, [contour], -1, (255, 255, 255), -1)
mask = cv2.bitwise_and(img, img, mask=mask)
```

Vpravo na obr. 5.4 je vykreslena násobená maska ohraničeným oblastí se vstupním obrazem. Vlevo je výsledná oblast ohraničená obdélníkovým útvarem a označena jako detekovaný objekt.

Obr. 5.3: Vlevo obrazový výstup detekce MSER, vpravo snímek vzniklý logickým součinem masky a vstupního snímku.

Nejnáročnější operací je funkce `mser.detectRegions()` jenž zabírá dvě třetiny výpočetního času pro jeden snímek (v tabulce 5.3). Při této metodě lze dosáhnout počtu 8 až 9 snímků za sekundu.

Tab. 5.3: Výpočetní náročnost metody MSER.

Proces	čas [ms]
Předzpracování snímku	19,616
Detekce MSER oblastí	74,428
Vykreslení oblasti s pohybem	17,220
Celkový čas pro jeden snímek	111,264

5.4 Algoritmus založený na metodě ORB

Metoda pracuje s obrazem za předpokladu, že objekty na obloze jsou detekovatelné pomocí klíčových bodů metody ORB. Pro inicializaci je použita funkce `cv2.ORB_create()`. Parametry jsou maximální počet získaných příznaků a úroveň obrazové pyramidy. Obrazová pyramida vzniká konvolucí s Gaussovým jádrem.

Výpis 5.6: Výpis kódu pro výpočet významných bodů metody ORB

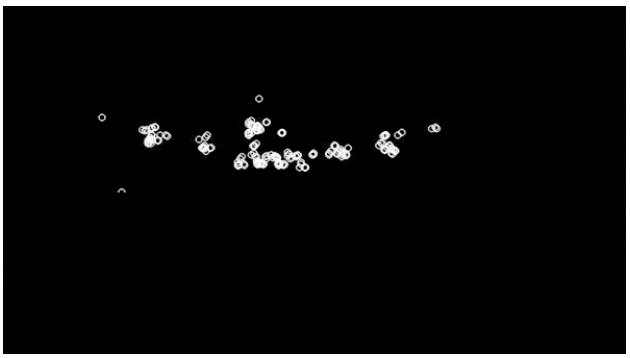
```
cap = cv2.VideoCapture("video.mp4")
orb = cv2.ORB_create(500,1.2)
while (1):
  ret, frame = cap.read()
  kp, des = orb.detect(frame, None)
```

V první fázi je načten snímek, ze kterého jsou pomocí `orb.detect()` nalezeny významné body. Tyto body jsou vykresleny do předpřipravené masky (na obr. 5.4), ve které jsou odstraněny nepodstatné oblasti detekce.

Výpis 5.7: Výpis kódu pro vykreslení významných bodů a jejich lokalizaci

```
mask = np.zeros((img1.shape[0], img1.shape[1], 1), dtype=np.uint8)
mask = cv2.drawKeypoints(mask, kp1, mask, color=(255, 255, 255))
cv2.rectangle(mask, (0, 200), (600, 480), (0, 0, 0), thickness=-1)
(_, cnts, _) = cv2.findContours(mask.copy())
```

Další část kódu je realizovaná obdobným způsobem jako ve výpise 5.2. Vykreslené významné ORB body, vlevo na obr. 5.5, jsou ohraničeny obdélníkovým útvarem, který je vykreslen do původního snímku.



Obr. 5.4: Vyznačené významné body na definované masce.

Obr. 5.5: Vlevo nalezené významné body ve videosnímku, vpravo detekovaný objekt

Výhodou oproti ostatním metodám je zejména v nezávislosti na pohybu kamery. Metoda je také odolná vůči mírné oblačnosti, příkladem je obr. 5.6. Mraky nevykazují podmínky pro lokalizaci významných bodů, a tak je lze lépe odlišit od objektů.

Obr. 5.6: Vlevo nahoře definovaná maska s vyznačenými body, vpravo nahoře všechny vyhledané body. Dole snímek s výslednou detekcí.

Z hlediska času potřebného pro výpočet, jednotlivé procesy rozepsány v tab. 5.4, je metoda ORB velmi výhodná. Metoda využívá vyhledávání významných oblastí za pomocí FAST detektoru, odpadá tedy výpočet deskriptoru, který je součástí metody ORB. Metoda slouží velmi efektivně k lokalizaci objektů, a to nezávisle na

pohybu kamery či osvětlení (viz obr. 5.6). Proces detekce dosahuje 10 až 11 snímků za sekundu.

Tab. 5.4: Srovnání časové náročnosti pro výpočet jednoho snímku metodou významných bodů ORB.

Proces	čas [ms]
Načtení a lokalizace bodů	51,978
Vykreslení bodů na masce	22,084
Vykreslení oblasti s pohybem	17,142
Celkový čas pro jeden snímek	91,204

5.5 Rozpoznávání objektů korelační metodou

Předpokladem pro rozpoznávání objektů na základě korelační metody je výběr vstupního snímku, případně více snímků, které budou sloužit jako šablona. Na základě těchto vstupních snímků předpokládáme, že se daný objekt v obraze vyskytuje a bude mít stejný tvar, velikost a úhel natočení. Zlepšení oproti klasické korelační metodě spočívá v použití snímků, které prošly Cannyho hranovým detektorem. Příkladem je snímek na obr. 5.7, kde se nacházejí hrany původního snímku. Pro proces korelace jsou tedy použity hranové mapy, jenž nezávisí na osvětlení snímané scény. Je tak možné s dostatečnou přesností detekovat shodně se opakovatelné pohyby. Nevýhodou je, že nelze rozpoznávat předměty z jiných úhlů a různých natočení.

Po načtení vstupního snímku a aktuálního snímku ve videu se provede převod do stupní šedi a následně proběhne Cannyho hranový detektor `cv2.Canny` s dvěma prahy pro hysterezní prahování (viz kapitola 2.3.1).

Výpis 5.8: Výpis kódu pro Cannyho hranový detektor

```
template = cv2.imread('plane.jpg')                                    1
template = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)                 2
template = cv2.Canny(template, 50, 200)                               3
```

Obr. 5.7: Model letadlo po aplikaci Cannyho hranového detektoru.

V další fázi je provedena korelace pomocí funkce `cv2.matchTemplate()`. Tímto vznikne korelační obraz (v pravé části obr. 5.8), ve kterém je následně lokalizováno maximum. Pokud toto maximum překročí určitý práh pro detekci je stanovena pozice tohoto maxima jako levý horní roh objektu. Další rozměry jsou dopočítány z rozměrů předlohy a ve snímaném obraze je tento objekt označen zeleným obdélníkem.

Výpis 5.9: Výpis kódu pro korelační metodu

```
res = cv2.matchTemplate(gray,template,cv2.TM_CCOEFF_NORMED)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
top_left = max_loc

if max_val > 0.138:
  bottom_right = (top_left[0] + w, top_left[1] + h)
  cv2.rectangle(frame, top_left, bottom_right, (0, 255, 0), 2)
```

Značnou nevýhodou této metody je nutnost předlohy, u které je nezbytné, aby měla stejné obrysy, natočení a velikost.

Obr. 5.8: Příklad použití korelační analýzy pro přistávající letadlo. Předloha na obr.5.7

Z důvodu výpočtu vzájemné korelace snímků je z hlediska výpočetní rychlosti (tab. 5.5) tato metoda o něco náročnější než předešlé. Výpočetní rychlost dosahuje 6 až 7 snímky za sekundu.

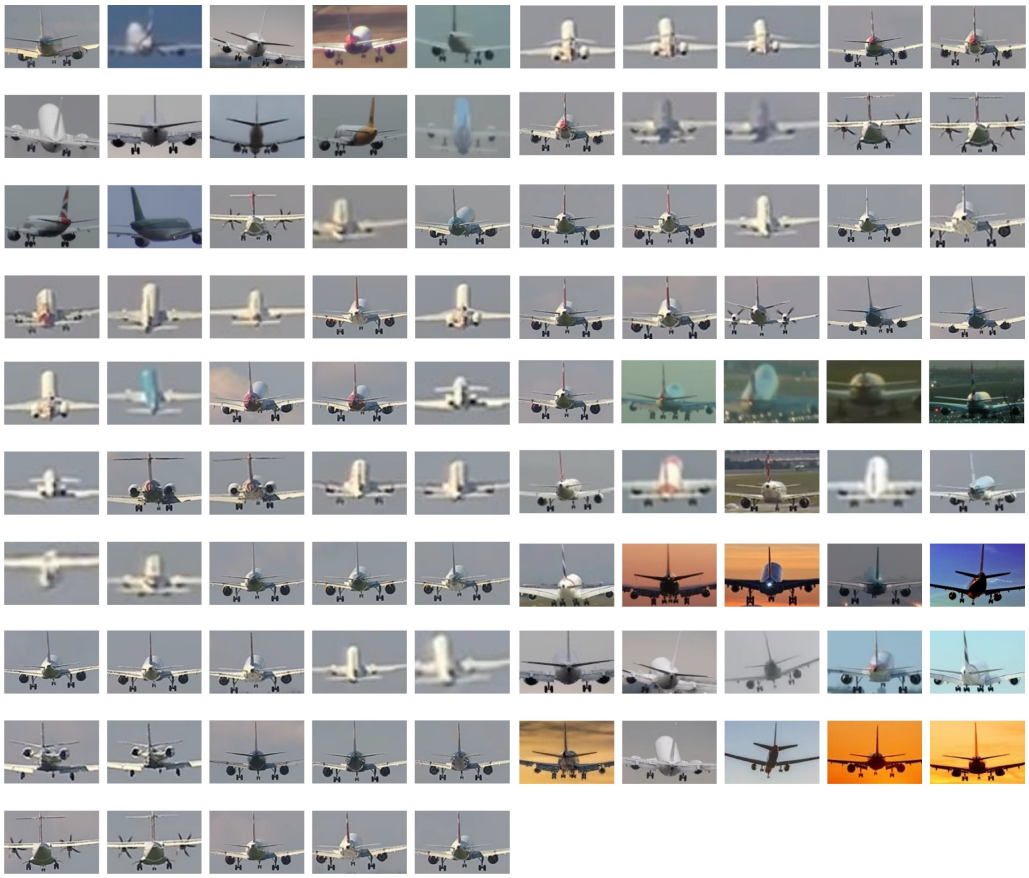
Tab. 5.5: Srovnání časové náročnosti pro výpočet jednoho snímku korelační metodou.

Proces	čas [ms]
Cannyho hranový detektor	43,856
Korelace s předlohou	99,633
Vykreslení oblasti s pohybem	13,562
Celkový čas pro jeden snímek	157,051

5.6 Rozpoznávání objektů na základě Haarových příznaků

Tato metoda zahrnuje použití tzv. klasifikátoru, jenž byl natrénován na základě vstupních snímků a aplikací Haarových příznaků. K účelu trénování je zapotřebí velké množství snímků, a to jak těch, na kterých se daný objekt vyskytuje, tak i negativních, které například reprezentují pozadí objektů. Negativní snímky by neměly obsahovat daný objekt a ani jeho části. Metoda je dále zmíněna v kapitole 4.5.

Čím více snímků, tím více je zatížen CPU a je potřeba větší výpočetní výkon. V tomto kroku bylo rozhodnuto, že samotné trénování není vhodné provést na platformě Raspberry Pi, z důvodu velké výpočetní náročnosti. Klasifikátor je tak předtrénován aplikací *„Cascade Trainer GUI“* [26]. Rozpoznávací algoritmus je o něco méně výkonově náročný, a je tak možné ho implementovat na Raspberry Pi, které s těmito daty pracuje.



Obr. 5.9: Sada snímků určená k natrénování.

Pro účely testování bylo vybráno téměř sto pozitivních a pět set negativních snímků. Podmínka k tomu, aby natrénovaný klasifikátor byl kvalitní není dána počtem snímků, spíše je důležité postihnout snímky z různých prostředí.

Zmíněný program *„Cascade Trainer GUI“* vytvoří za pomocí trénovací sady v několika fázích silný klasifikátor. Po zadání velikosti vstupních dat a výběru typů příznaků následuje výstup v podobě klasifikátoru v souboru s příponou *.xml*.

Po načtení vstupního videa a předtrénovaného klasifikátoru jsou objekty detekovány pomocí funkce `cv2.detectMultiScale()`. Parametrem je klasifikátor, vstupní obraz, měřítko, počet sousedních oblastí, minimální a maximální velikost detekovaného okna. [8] Vhodným nastavením lze docílit požadového výstupu detekce.

Výpis 5.10: Výpis kódu pro detekci pomocí Haarových příznaků

```
cascade = cv2.CascadeClassifier('cascade.xml')                    1
rects = cascade.detectMultiScale(gray, 1.1,10)                    2
```

Algoritmus pro rozpoznávání objektů dosahuje rychlosti 157,520 ms za jeden snímek, to znamená mezi 6 a 7 snímky za sekundu. V rámci testování byla také sledována úspěšnost detekce. V tabulce 5.6 jsou shrnuty výsledky, a to bez průměrování a s průměrováním detekčních oken, které vedou k zajištění větší stability algoritmu (dále zmíněno v podkapitole 6.3.4)

Tab. 5.6: Úspěšnosti kaskádního klasifikátoru před a po použití průměrovacího algoritmu

scale factor	bez průměrování	s průměrováním
1,2	22,14 %	43,6 %
1,1	60,7 %	77,8 %
1,05	94,3 %	95,7 %

6 PRAKTICKÁ REALIZACE

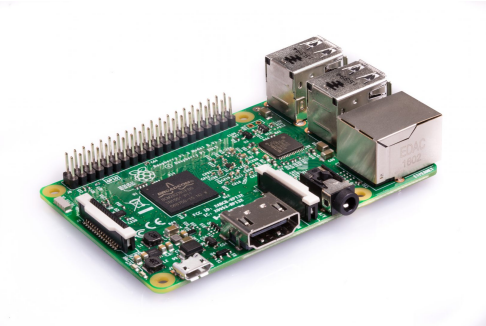
Tato část práce se zabývá implementačními nástroji pro detekci pohybujících se objektů. Popisuje hardwarové a softwarové prostředky a realizaci algoritmů.

6.1 Hardwarové prostředky

V této podkapitole je popsán veškerý hardware pro realizaci jednotlivých algoritmů.

6.1.1 Raspberry Pi

Raspberry Pi třetí generace je populární miniaturizovaný mikropočítač založený na rodině architektury ARM.



Obr. 6.1: Raspberry Pi 3 Model B [12]

Pro praktickou realizaci byl vybrán konkrétně mikropočítač Raspberry Pi 3B+ (na obr. 6.1), a to z mnoha důvodů. V současné době se jedná o nejvýkonnější model tohoto mikropočítače. V porovnání s předešlými modely, jejichž parametry jsou uvedeny v tab. 6.1, tato verze nabízí integrovaný WiFi a Bluetooth modul. Procesor Cortex A7 byl nahrazena jádrem Cortex-A53 s 64-bitovou architekturou. Nezbytnou součástí je i přítomnost unixového operačního systému Raspbian, který je optimalizovaný pro procesory ARM.

Širokou škálu využití nabízí i díky svým periferiím. Na desce se nacházejí 4 USB konektory, audio 3,5mm jack, 40 GPIO pinů, HDMI konektor, DSI konektor, micro USB socket k napájení a dva pátnáctipinové konektory, které slouží k připojení kamery a displeje pomocí flex kabelu.

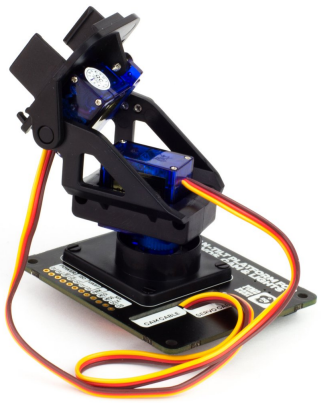
Raspberry Pi nabízí nespočet množství praktického využití, lze jej použít jako prvek inteligentních domácností, kamerový sledovací systém a nebo jako webserver. [21]

Tab. 6.1: Přehledová tabulka parametrů mikropočítačů Raspberry Pi. [21]

Název	SoC	Architektura	Počet jader	Frekvence CPU	RAM
Raspberry Pi Zero	BCM2835	ARM11	1	1,0 Ghz	512 MB
Raspberry Pi B	BCM2835	ARM 11	1	700 MHz	512 MB
Raspberry Pi 2B	BCM2837	ARM Cortex-A53	4	900 Mhz	1 GB
Raspberry Pi 3B	BCM2836	ARM Cortex-A7	4	1,2 Ghz	1 GB

6.1.2 Otočný modul s kamerou

K vybranému mikropočítači Raspberry Pi 3B+ se nabízí možnost připojit buď klasickou webkameru pomocí USB konektoru, nebo přímo oficiální PiCameru za pomocí CSI rozhraní a flex kabelu. Pro tuto práci byla vybrána PiCamera a doplněna o modul Pan-Tilt HAT, který zajišťuje pomocí 2 servomotorů natáčení v rozsahu 180 stupňů. Modul Pan-Tilt HAT dále umožňuje připojení přisvětlovacích LED diod. Toto zařízení komunikuje přes I2C rozhraní a využívá knihovny Pantilthat. Otočný modul tak zajišťuje při dynamickém režimu otáčení kamery a sledování objektu. [30]



Obr. 6.2: Otočný modul Pantilt-HAT [30]

6.2 Softwarové prostředky

V této podkapitole je popsán programovací jazyk Python, jeho výhody a také použité rozšiřující balíčky pro práci s obrazem.

6.2.1 Python

Python je interpretovaný skriptovací jazyk. Jedná se o open source projekt, který je nabízen zdarma na řadě běžných platforem (Unix, Windows, macOs), ve většině distribucí GNU/Linux včetně Raspbianu je Python součástí základní instalace. [25]

Má dobře čitelnou syntaxi, je objektově orientovaný a disponuje širokou uživatelskou základnou. Dále zahrnuje bohatou škálu knihoven, s jejichž pomocí můžeme řešit i zdánlivě složité úkoly, které bychom např. ve vědeckém SW jako je Matlab řešili velmi obtížně. Může jít o zpracování dat z netradičního formátu, webové rozhraní a také ovládání periferie. Mimo jiné v Pythonu nalezneme i knihovnu OpenCV, která vznikla za účelem řešení problémů počítačového vidění. [23]

6.2.2 Knihovna OpenCV

Knihovna OpenCV vznikla jako otevřená multiplatformní knihovna pro zpracování obrazu. Podporuje řadu programovacích jazyků jako je C/C++, Python, Java a je dostupná pod různými operačními systémy (Windows, Linux, OS X, Android či iOS). [1] Knihovna byla navrhnuta s velkou výpočetní efektivitou a se zaměřením na aplikace probíhající v reálném čase. V současnosti nabízí přes 500 funkcí, které uživateli poskytují rozmanité možnosti aplikovat sofistikované algoritmy pro analýzu obrazu. [24]

Instalace knihovny OpenCV

Pro instalaci knihovny OpenCV je v tomto zvolen operační systém Raspbian Strech. Zaprvé je třeba nainstalovat vývojářský nástroj `CMake`, který později konfiguruje proces a finální kompilaci knihovny OpenCV. Nezbytnou součástí je též instalace balíčků pro čtení obrazu a dat.

Výpis 6.1: Instalace nezbytných balíčků a CMake pro instalaci knihovny OpenCV

```
$ sudo apt-get install build-essential cmake pkg-config                1
$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev          2
```

```
$ sudo apt-get install libpng12-dev libavcodec-dev libavformat-dev
$ sudo apt-get install libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
```

Další podmínkou je instalace balíčku `GTK`, který umožňuje vytvářet základní grafické rozhraní.

Výpis 6.2: Instalace nezbytných balíčků pro práci s knihovnou OpenCV

```
$ sudo apt-get install libgtk2.0-dev libgtk-3-dev
$ sudo apt-get install libatlas-base-dev gfortran
```

Jsou-li nezbytné balíčky úspěšně nainstalovány, pokračujeme stáhnutím knihovny OpenCV z oficiálního repozitáře.

Výpis 6.3: Stáhnutí knihovny OpenCV z oficiálního repozitáře

```
$ cd ~
$ wget opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip
$ unzip opencv.zip
```

Nyní je vše připravené ke zkompilování a instalaci. V dalším kroku založíme složku, do které bude umístěna zkompilovaná knihovna OpenCV.

Výpis 6.4: Kompilace a instalace knihovny OpenCV

```
$ cd ~/opencv-3.3.0/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules \
-D BUILD_EXAMPLES=ON ..

$ make -j4
```

Doba kompilace trvá kolem 2 hodin. Po úspěšné kompilaci je možné finálně knihovnu OpenCV nainstalovat, a to následujícím příkazem.[31]

Výpis 6.5: Finální instalace knihovny OpenCV

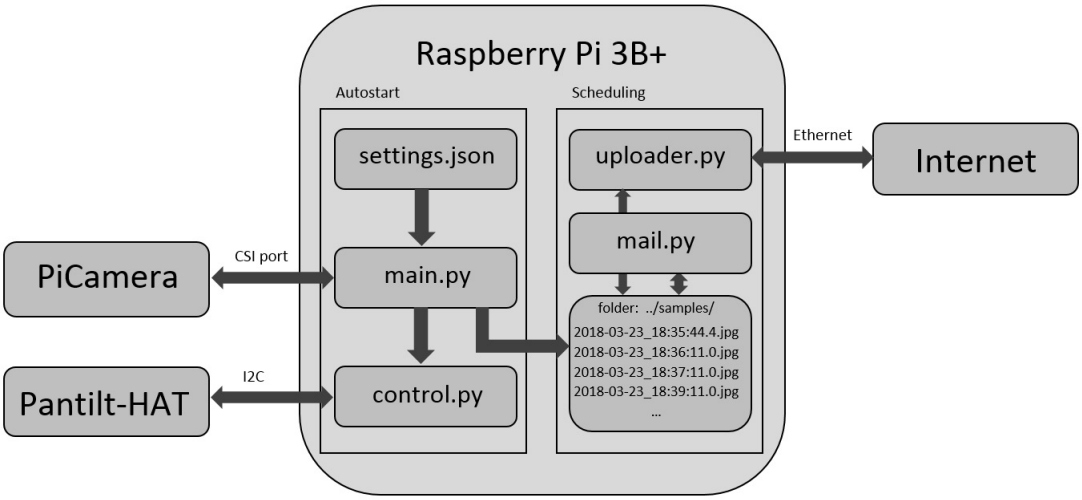
```
$ sudo make install
$ sudo ldconfig
```

Nainstalovaná knihovna OpenCV je nyní v Pythonu dostupná pod příkazem `import cv2`.

6.3 Realizace zařízení pro detekci objektů

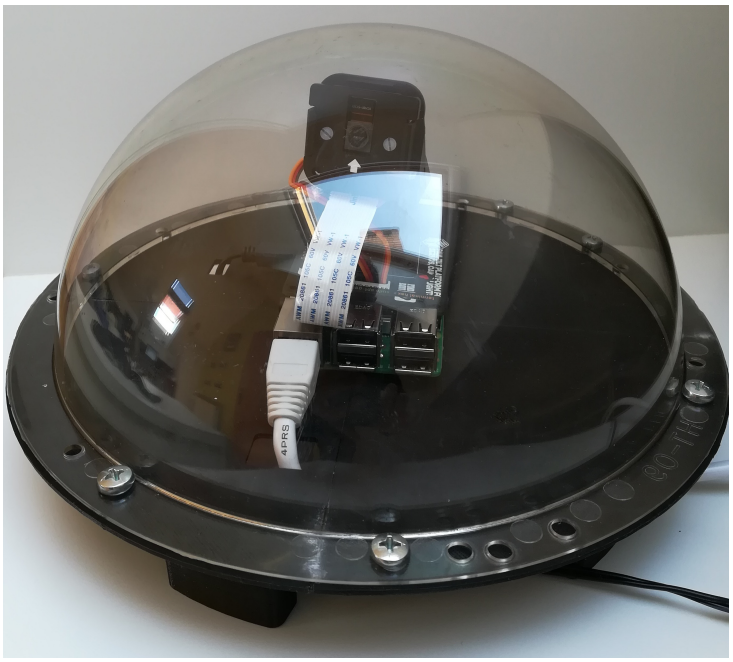
Pro realizaci byl zvolen mikropočítač Raspberry Pi 3B+. Na obrázku 6.3 je zobrazeno schéma navrženého detekčního zařízení. Primární úlohu zastupuje mikropočítač, který po připojení napájení nabootuje operační systém a spustí detekční program. Celý algoritmus je rozdělen na dílčí podprogramy. Část skriptů běží automaticky od spuštění zařízení a část je časovaná a spouští skripty v definovaných intervalech.

Pro detekci objektů slouží `main.py`, který zpracovává obraz podle zvoleného nastavení z konfiguračního souboru `settings.json`. Součástí je dále `control.py`, který zajišťuje ovládání otáčecího modulu Pantilt-HAT. Detekované snímky jsou uloženy do složky `../samples/`, s kterou pracují skripty `uploader.py` a `mail.py`. Tyto skripty v minutových intervalech kontrolují obsah složky s detekovanými objekty a zajišťují odesílání upozornění a nahrávání snímků na cloudové úložiště. Pokud jsou snímky nahrány na server, pak jsou smazány v lokální složce.



Obr. 6.3: Schéma zařízení pro detekci objektů

Finální podoba zařízení je na obr. 6.4. Spodní díl byl vyroben na 3D tiskárně. Kopule zakrývá otočné zařízení s kamerou a Raspberry Pi.



Obr. 6.4: Finální podoba zařízení pro detekci objektů.

6.3.1 Automatické spuštění a časování skriptů

Je-li třeba zajistit automatické spouštění nebo provedení nějakého příkazu, pak lze použít skript `rc.local`, který se spouští po startu zařízení. Toho lze využít zejména v případě, kdy Raspberry Pi pracuje bez připojených periferiích, jenž by umožňovaly manuální spuštění detekčního algoritmu.

Výpis 6.6: Příkaz ke spuštění skriptu rc.local

```
sudo nano /etc/rc.local
```

Jak je ze schématu na obr. 6.3 patrné, do skriptu `rc.local` jsou uloženy cesty ke spuštění hlavního detekčního algoritmu `main.py` a skriptu zajišťující otáčení kamery `control.py`.

Další skupina podprogramů slouží k odesílání upozornění na e-mail a k nahrávání na cloudové úložiště Dropbox. Tyto skripty neprobíhají v nekonečné smyčce, ale pouze jednou za určitý interval a pravidelně kontrolují obsah složky s detekovanými snímky. Po dokončení procesu vedoucí k nahrání snímků na server se automaticky skript ukončí. Pokud není dostupné připojení jsou ukončeny ihned. Jejich spuštění je zajištěno nástrojem `Cron`, který automaticky plánuje čas spouštění skriptů podle nastavených intervalů. Automatické spouštění lze nastavit ve zvolený den v týdnu,

den v měsíci, měsíc, hodinu a minutu. Pro opakovaní za každou časovou jednotku je definována hvězdička. Konkrétní nastavení pro skript `mail.py` a `control.py` se nachází v následujícím výpise6.7.

Výpis 6.7: Cron skript k časování detekčních podprogramů

```
# dow - day of week (0 - 7)
# mon - month (1 - 12)
# dow - day of month (1 - 31)
# h - hour (0 - 23)
# m - min (0 - 59)
# m h dom mon dow command
  * *  *   *   *  python3 /home/pi/Dropbox-Uploader/uploader.py
  * *  *   *   *  python3 /home/pi/home/pi/bcprace/mail.py
```

6.3.2 Funkce algoritmu

Konfigurační soubor `settings.json` (ve výpise 6.8) umožňuje nastavení parametrů pro detekci. Lze nastavit, zda má být použita kamera či jiný zdroj videa, případně cesta k uloženému videu. Mezi další možnosti patří výběr, zda snímání bude pouze statické nebo i dynamické, kdy se kamera otáčí za sledovaným objektem.

Statický režim využívá pro detekci nežádoucích objektů metodu subtrakce pozadí. Tato metoda je vhodná pro použití na staticky upevněnou kameru. V případě dynamického režimu se objekty detekují na základě metody významných bodů ORB. Detekce tak probíhá za pohybu kamery. Mimo výběr metody lze povolit detekci horizontu, která umožňuje zapnout či vypnout detekci objektů pod horizontem.

Výpis 6.8: Struktura nastavení parametrů pro detekci objektů

```
{
    "detection":
    {
        "source": "video",
        "path":"/home/pi/home/pi/bcprace/drone.mpg",
        "config": "static",
        "horizont": 1
    },


    "settings":
    {
        "position":
        {
            "x":0,
            "y":0
```

```
        },                                                                  17
        "points": 500,                                                      18
        "timedetect":"5",                                                   19
        "minsize":"10"                                                      20
    },                                                                      21
                                                                            22
    "classifier":                                                           23
    {                                                                       24
        "allow":"1",                                                        25
        "path":"/home/pi/home/pi/bcprace/airplane.xml",                     26
        "sensitivity":"1.3"                                                 27
    }                                                                       28
}                                                                           29
```

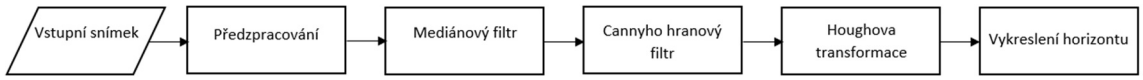
Pomocí proměnných x, y lze předem definovat počáteční polohu kamery. Tyto hodnoty reprezentují úhel otočného modulu, který je možný otáčet v rozmezí od -90° do 90°, a to vertikálně i horizontálně.

Jestliže má být pro rozpoznávání natrénovaných objektů použit klasifikátor, pak je nezbytné doplnit přístupovou cestu k souboru .xml s natrénovaným klasifikátorem. Jednou z možností, jak dále měnit citlivost rozpoznávání objektů, je změnit měřítku pro detekci (v rozsahu 1-2).

Pro požadované vyhodnocení detekce je dále možné nastavit minimální velikost detekčního okna a minimální počet kontinuálních snímků, na nichž je detekovaný objekt. Tyto parametry slouží k nastavení upozorňovacího mechanismu, aby nedocházelo k zasíláním zbytečně velkého množství snímků.

6.3.3 Detekce horizontu

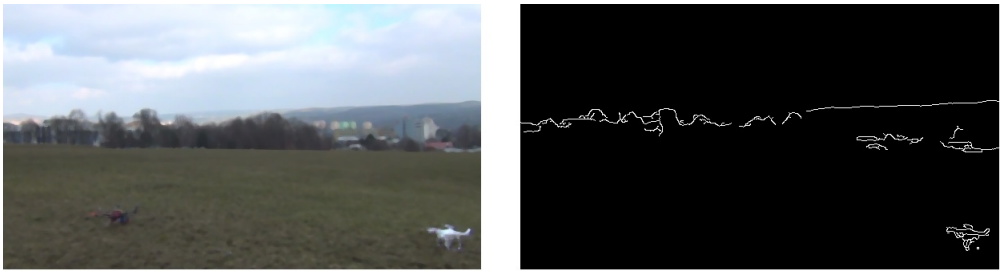
Po fázi načtení konfiguračního souboru a vstupního videa je v prvním kroku detekován horizont. Toho je docíleno pomocí Cannyho hranového filtru a Houghovy transformace. Vývojový diagram je popsán obrázkem 6.5.



Obr. 6.5: Vývojový diagram pro detekci horizontu

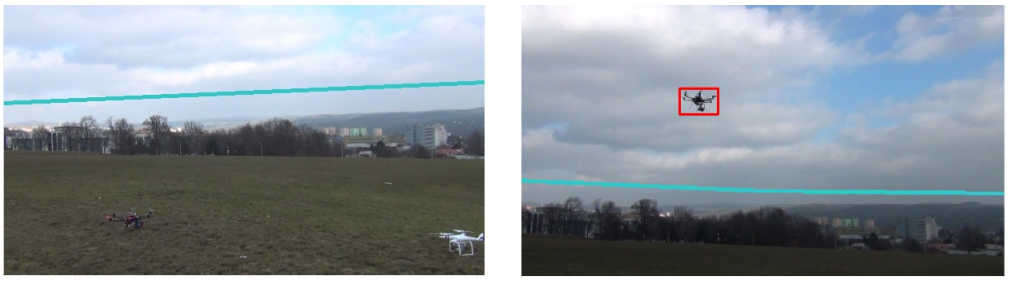
Po načtení vstupního snímku je na obraz aplikovan mediánový filtr (vlevo na obr. 6.6), který odstraňuje náhodný šum. Výstupem Cannyho hranového filtru je

černobílý obraz, který indikuje hranové pixely (vpravo na obr. 6.6).



Obr. 6.6: Vlevo vstupní obraz po aplikaci mediánového filtru, vpravo po použití Cannyho hranového filtru

Houghova transformace poté nalezne nejvýraznější přímku ve snímku. Předpokládame, že tato přímka znázorňuje horizont. Jestliže je známa poloha horizontu, pak jsou detekovány všechny objekty nad touto přímkou (na obr. 6.8).



Obr. 6.7: Příklad snímků s vykreslenou přímkou horizontu

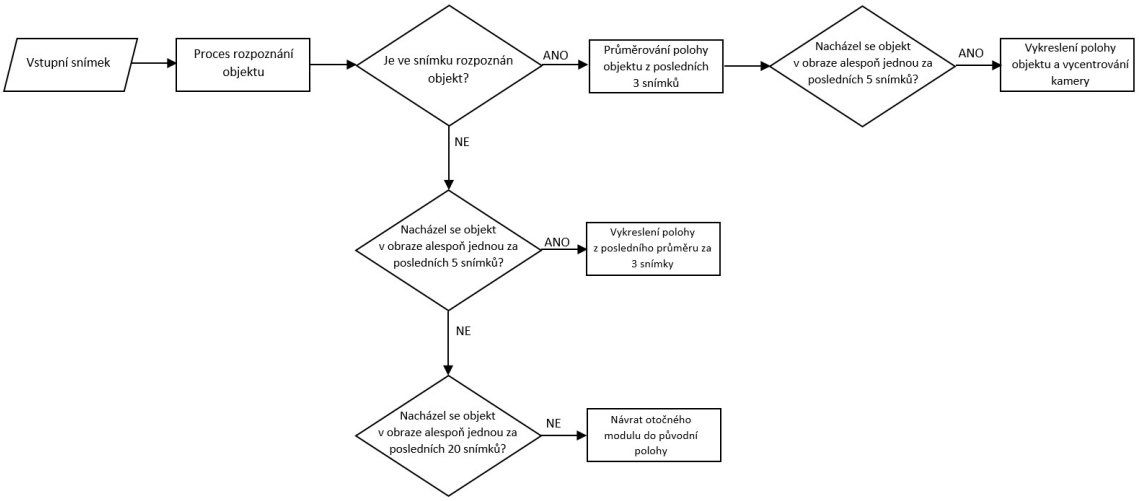
Oblast pod přímkou je maskována a ve výsledné detekci se neuvažuje. Pokud přesto konkrétní aplikace vyžaduje, aby byly v obraze nalezeny veškeré objekty, pak lze detekci horizontu v konfiguračním souboru vypnout.

6.3.4 Rozpoznávání a sledování objektu

Algoritmus umožňuje rozpoznat objekty, které jsou předem natrénované pomocí Haarových příznaků a kaskádního klasifikátoru. Tento princip byl již vysvětlen v předchozích kapitolách 4.5 a 5.6. Vývojový diagram pro zajištění rozpoznání objektu a jeho sledování je znázorněno na obr. 6.8

Po načtení vstupního snímku prochází každý snímek procesem rozpoznávání na základě kaskádního klasifikátoru z předem natrénované sady dat. Pokud je detekován objekt, pak prochází vždy daná oblast za poslední tři snímky průměrovacím mechanismem, aby byla zajištěna stabilita označeného okna. Výsledky testování tohoto algoritmu jsou uvedeny v podkapitole 5.6. Jestliže se okno nachází jinde než na středu záběru, pak dochází k vycentrování otočného modulu na střed označeného okna s objektem.

Pokud je objekt ztracen, nebo je mimo rozsah otáčecího modulu, pak detekční okno setrvá dalších 5 snímků dokud nezmizí. To je z důvodu, aby se nedocházelo ihned v přerušení při neúspěšném rozpoznání objektu. Otáčecí modul setrvá v místě, kde byl naposledy detekován tento objekt, do doby dalších zpracování 20 snímků a to pro případ, že se na scéně objekt ještě vyskytne. Po uplynutí této doby je kamera navrácena do původní polohy.



Obr. 6.8: Vývojový diagram pro rozpoznávání objektů

6.3.5 Detekce nežádoucích objektů

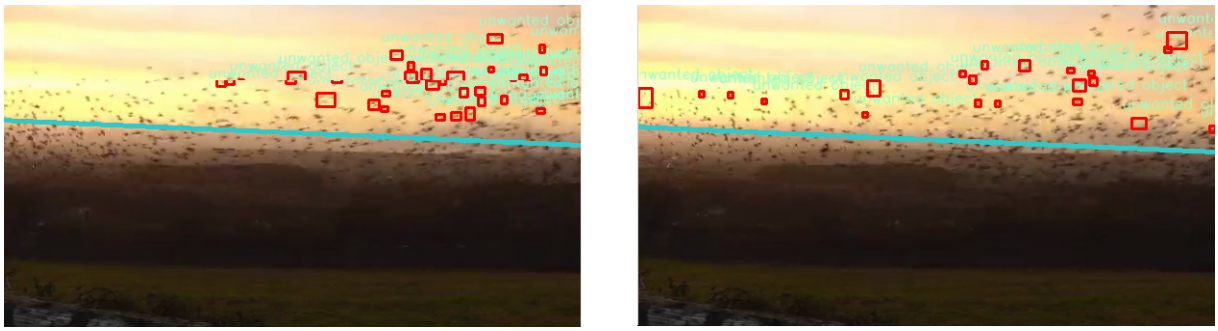
Pro detekci objektů, jako jsou ptáci či drony na obloze, je možné použít více metod. Tyto přístupy lze dělit na použití ve statickém nebo dynamickém módu.

Pokud se kamera nepohybuje, pak je nejefektivnější použít metodu subtrakce pozadí. Tato metoda je založeno na principu diferenční metody, kdy se od sebe odečítá aktuální snímek a snímek pozadí. Obraz pozadí je však průběžně aktualizován, kdy s každým dalším snímkem je k obrazu pozadí přičítán i aktuální snímek,

a to s určitou vahou. Tímto způsobem lze eliminovat chybnou detekci způsobenou neaktuálním snímkem pozadí.

Výpis 6.9: Výpis kódu pro funkci, která aktualizuje snímek pozadí

```
def update_BG(alpha, frame, backGroundModel):
  backGroundModel = frame*alpha + backGroundModel*(1 - alpha)
  return backGroundModel.astype(np.uint8)
```



Obr. 6.9: Okno s označenými objekty pro statický režim

V případě dynamického režimu se předpokládá, že se kamera bude při detekci pohybovat. Pro tento účel je použita metoda detekce významných bodů ORB. Detekované body jsou nalezeny v každém snímku. Jejich počet je možné shora omezit v nastavení konfiguračního souboru. Množina všech nalezených bodů ORB je vykreslena na čistou masku. Každá tato oblast je dále testována Cannyho hranovým filtrem. Pokud je objekt detekován jako nežádoucí, pak je označen červeným detekčním oknem. Metodou dvojitého ověření lze předejít četným výskytům falešné detekce, která může nastat při větší oblačnosti.

Obr. 6.10: Okno s označenými objekty pro dynamický režim

6.3.6 Odesílání snímků na cloud a mailové upozornění

V případě nalezení nežádoucího objektu, prochází daný snímek dalším testem, aby se eliminovalo duplicitní odesílání snímku se stejným objektem. V každém snímku je zjištěno, kolik se v obraze vyskytuje nežádoucích objektů. Pokud nastane jakákoliv změna v počtu objektů, pak je snímek zaznamenán. Všechny snímky, které úspěšně projdou tímto testem jsou uloženy.

K nahrávání snímků na cloudové úložiště slouží skript `mail.py`, který je automaticky spouštěn za určitý časový interval (v tomto případě 1 minuta) a kontroluje složku obsahující s detekovanými snímky. V případě, že se ve složce vyskytují snímky, následuje jejich nahrání na cloudové úložiště. Po úspěšném nahrání konkrétních snímků, jsou ty stejné v lokální složce smazány a snímky jsou dostupné pouze ze serverového úložiště.

Podobně jako nahrávání snímků je řešeno i odesílání upozornění na nově detekované objekty. Odeslání mailového upozornění zahrnuje údaj o počtu detekovaných objektů a čas kdy byly detekovány.

6.4 Zhodnocení detekčních metod

Závěrečné zhodnocení shrnuje problémy dílčích algoritmů a jejich výpočetní náročnost. Úspěšnost detekce je však závislá na mnoha subjektivních vlastnostech, jako je oblačnost, osvětlení, srážky a podobně. Nejobjektivnějším měřítkem pro jejich srovnání je jejich výpočetní náročnost, uvedena v tabulce 6.2.

Tab. 6.2: Výpočetní náročnost pro jednotlivé procesy

Proces	Minimum	Maximum	Průměr
Předzpracování	116 ms	200 ms	150 ms
Detekce horizontu	27 ms	119 ms	50 ms
Rozpoznávání objektů	52 ms	55 ms	53 ms
Detekce objektů	151 ms	250 ms	200 ms
Celkem	348 ms	624 ms	453 ms

Jednotlivé procesy trvají různou dobu, a to z důvodu rozvětveného kódu. Závisí tedy zejména na počtu detekovaných objektů a na nastavení detekce. Dalším kritériem pro výpočetní rychlost je rozlišení obrazu, které je 600x450 pixelů. Rychlost

zpracování se pohybuje mezi 2 až 3 snímky za sekundu, což je vzhledem k předpokládanému účelu dostatečné.

Při detekci horizontu nastávají chybné detekce z principu metody, kdy v obraze je nalezeno více rovných čar. Nejvíce chyb vzniká, jestliže se zařízení umístí v blízkosti budov, či jiných nerovností v obraze (obr. 6.11). Dalším problém je mlha a příliš nevýrazný horizont. Úspěšnost detekce tedy silně závisí na vstupním obraze.

Obr. 6.11: Chyby vzniklé při nesprávném určení horizontu

Je-li nesprávně vyhodnocena poloha horizontu, dochází i k chybám při detekci nežádoucích objektů. Další chybná detekce nastává v případě kupovité oblačnosti (vlevo na obr. 6.12).

Obr. 6.12: Odolnost detekční metody vůči oblačnosti, vlevo se vyskytuje chybná detekce při kraji mraku.

V rámci testování tak bylo zjištěno, že detekčních algoritmy vyhodnocují objekt bezchybně pouze na bezmračné obloze. Algoritmus je odolný vůči mírné oblačnosti, nejvíce chyb pak vzniká při okrajích kupovité oblačnosti. Některé náhodné chyby lze odstranit metodou dvojího ověření, zmíněnou v kapitole 6.3.5

7 ZÁVĚR

Tato bakalářská práce se zabývala problematikou detekce pohybujících se objektů. Cílem práce bylo vytvořit algoritmus, který by byl schopný detekovat a rozlišit objekty na obloze.

První část práce byla přehledem metod vhodných pro detekci objektů a jejich rozpoznávání. Z detekce pohybu se ve finálním zařízení uplatňuje metoda subtrakce pozadí, která je ideální pro detekci statického obrazu. Naopak pro zvolený dynamický režim je použita metoda detekce významných bodů ORB, která byla ověřena zejména na odolnost vůči chybám vzniklých při oblačnosti. Jak ve statickém, tak i v dynamickém režimu lze detekovat horizont, který odstraňuje všechny objekty nacházející se pod přímkou horizontu. Důvodem je eliminace chybných detekcí při zemi. V případě, že se na obloze vyskytuje předem definovaný objekt (letadlo), pak je kaskádním klasifikátorem detekován a je vycentrován na střed záběru. Pokud dojde k detekci jakéhokoliv nežádoucího objektu, pak je celý snímek uložen, nahrán na cloudové úložiště a je zasláno upozornění.

V rámci této práce proběhlo testování jednotlivých částí algoritmu. Testovala se výpočetní náročnost a úspěšnost detekce. Doba zpracování celého algoritmu se pohybuje od dvou do tří snímků za sekundu a záleží na zvoleném nastavení a počtu detekovaných objektů. Zároveň byl v práci vylepšen kaskádní klasifikátor o průměrovací mechanismus a srovnána jeho úspěšnost. Největším zjištěným problémem pro detekci nežádoucích objektů byla oblačnost. Částečně lze tuto chybu vyloučit metodou dvojitého testu, kdy se obraz testuje na přítomnost hrany v obraze. I přesto jsou chybně detekovány kontrastní mraky na obloze při kupovité oblačnosti. Většina těchto chybných detekcí je velmi krátkých, a proto je možné je eliminovat nastavením minimální doby výskytu.

V praktické části práce bylo realizováno automatické detekční zařízení. Model zařízení se skládá z otočného modulu spolu s kamerou a Raspberry Pi, umístěných v průhledné kopuli s podstavcem vyrobeným na 3D tiskárně.

LITERATURA

[1] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image processing analysis and machine vision*. London: Chapman Hall, c1993. Chapman Hall Computing Series. ISBN 0412455706.

[2] HUANG, T. S. *Computer Vision: Evolution and Promise* [online]. [cit. 2017-10-16]. Dostupné z: http://cds.cern.ch/record/400313/files/p21.pdf

[3] Introduction to Image Processing. *Engineers Garage* [online]. [cit. 2017-10-24]. Dostupné z: https://www.engineersgarage.com/articles/image-processing-tutorial-applications?page=1

[4] NIXON, Mark S. a Alberto S AGUADO. *Feature extraction and image processin*. 2nd ed. Amsterdam: Academic Press, 2008. ISBN 978-0-12372-538-7.

[5] *OpenCV-Python Tutorials* [online]. 2013 [cit. 2017-11-21]. Dostupné z: https://opencv-python-tutroals.readthedocs.io/en/latest/

[6] Graphing a Gaussian Kernel in 3-D. *James D. McCaffrey* [online]. [cit. 2017-10-25]. Dostupné z: https://jamesmccaffrey.wordpress.com/2014/01/25/graphing-the-gaussian-kernel-function-in-3-d/

[7] *Získávání a analýza obrazové informace: Operace s obrazem II* [online]. , 15 [cit. 2017-10-25]. Dostupné z: https://www.med.muni.cz/biofyz/Image/prezentace_4.pdf

[8] *OpenCV 2.4.13.4 documentation* [online]. [cit. 2017-11-21]. Dostupné z: https://docs.opencv.org/2.4/

[9] SASTRASINH, Paul. Dense Realtime Optical Flow on the GPU. *Computer Science at Brown University* [online]. [cit. 2017-11-03]. Dostupné z: http://cs.brown.edu/courses/csci1290/2011/results/final/psastras/

[10] JOSHI, Prateek. *OpenCV with Python By Example*. ISBN 1785283936.

[11] Computer Science: Convert HSV to RGB colors. *Stackexchange.com* [online]. 2017 [cit. 2017-10-17]. Dostupné z: https://cs.stackexchange.com/questions/64549/convert-hsv-to-rgb-colors

[12] COLLINS, Robert. *Correspondence Matching* [online]. [cit. 2017-11-03]. Dostupné z: http://www.cse.psu.edu/ rtc12/CSE486/lecture07.pdf. Penn State University of Engineering.

[13] *Object Recognition and Template Matching* [online]. In: YANG, Ruigang. University of Kentucky, s. 23 [cit. 2017-11-03]. Dostupné z: http://www.vis.uky.edu/ ryang/Teaching/cs635-2016spring/Lectures/17-recognition.pdf

[14] *Interest point detection* [online]. [cit. 2017-07-04]. Dostupné z: https://en.wikipedia.org/wiki/Interest_point_detection

[15] BAY, Herbert, Andreas ESS, Tinne TUYTELAARS a Luc VAN GOOL. *Speeded-Up Robust Features (SURF)* [online]. 2008, , 14 [cit. 2017-08-12]. DOI: Herbert Bay a , Andreas Ess a , Tinne Tuytelaars b , and Luc Van Gool a,b. Dostupné z: ftp://ftp.vision.ee.ethz.ch/publications/articles/eth_biwi_00517.pdf

[16] HRÚZ, Marek. *SIFT, SURF, MSER* [online]. Plzeň, 2015 [cit. 2017-07-07]. Dostupné z: http://www.kky.zcu.cz/uploads/courses/mpv/04/materialy04.pdf

[17] XIE, Songyun, Wanpeng ZHANG, Wang YING a Khalid ZAKIM. *Fast Detecting Moving Objects in Moving Background using ORB: Feature Matching* [online]. Beijing, 2013, , 6 [cit. 2017-11-06]. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6568087tag=1

[18] ROSTEN, Edward. *FAST Corner Detection* [online]. 2015 [cit. 2017-11-06]. Dostupné z: https://www.edwardrosten.com/work/fast.html

[19] Research Gate: A maximally stable extremal regions system-on-chip for real-time visual surveillance. *Https://www.researchgate.net* [online]. 2015 [cit. 2017-11-02]. Dostupné z: https://www.researchgate.net/publication/304412048_A_maximally_stable_extremal_regions_system-on-chip_for_real-time_visual_surveillance

[20] CEN, Kaiqi. *Study of Viola-Jones Real Time Face Detector* [online]. , 5 [cit. 2017-09-18]. Dostupné z: https://web.stanford.edu/class/cs231a/prev_projects_2016/cs231a_final_report.pdf

[21] *Oficiální web Raspberry Pi* [online]. [cit. 2017-09-30]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[22] *Python* [online]. [cit. 2017-11-07]. Dostupné z: https://cs.wikipedia.org/wiki/Python

[23] *Proč Python?* [online]. In: URBAN, Jakub a Jan PÍPEK. 2013 [cit. 2017-11-07]. Dostupné z: http://pythonic.eu/fjfi/posts/proc-python.html

[24] HOWSE, Joseph. *OpenCV computer vision with Python: learn to capture videos, manipulate images, and track objects with Python using the OpenCV library.* Birmingham, England: Packt Publishing, 2013.

[25] *Python* [online]. [cit. 2017-11-07]. Dostupné z: https://cs.wikipedia.org/wiki/Python

[26] *CASCADE TRAINER GUI* [online]. [cit. 2017-11-28]. Dostupné z: http://amin-ahmadi.com/cascade-trainer-gui/

[27] Hough Line Transform. *OpenCV 3.0.0-dev documentation* [online]. [cit. 2018-04-01]. Dostupné z: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html

[28] Hough Transform. *Image processing learning resources* [online]. [cit. 2018-04-01]. Dostupné z: https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm

[29] *OpenCV 3.3.0 documentation* [online]. [cit. 2018-05-06]. Dostupné z: https://docs.opencv.org/3.3.0

[30] *PIMORONI: Pan-Tilt HAT* [online]. [cit. 2018-05-06]. Dostupné z: https://shop.pimoroni.com/products/pan-tilt-hat

[31] *Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi* [online]. 2017 [cit. 2018-05-08]. Dostupné z: https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ARM	Advanced RISC Machines
BRIEF	Binary Robust Independent Elementary Features
CPU	Central processor unit
DSI	Digital Serial Interface
FAST	Features from accelerated segment test
GNU	GNU's Not Unix!
GPIO	General-purpose input/output
GUI	Graphical user interface
HDMI	High-Definition Multimedia Interface
HSV	Hue, Saturation, Value
LBP	Local binary patterns
MSER	Maximally stable extremal regions
OpenCV	Open Source Computer Vision Library
ORB	Oriented FAST and Rotated BRIEF
px	pixel
RGB	Red, Green, Blue
SIFT	Scale Invariant Feature Transform
SoC	System on a chip
SURF	Speeded up robust features

A OBSAH PŘILOŽENÉHO CD

Na přiloženém médiu naleznete veškeré zdrojové kódy k programu. Pro úspěšné spuštění je třeba soubory přenést na kompatibilní zařízení s nainstalovými knihovnami OpenCV a Pantilt-HAT. Detekční algoritmus se nachází v souboru main.py. Nezbytné je nastavit parametry detekce a to v souboru settings.json. Pro spuštění je možné použít skript autorun.sh. Další skripty obsahují kódy zajišťující nahrávání na vzdálené úložiště a odesílání upozornění na e-mail.

- `/` ..... kořenový adresář přiloženého CD
  - `autorun.sh` ..... skript k automatickému spuštění algoritmu
  - `cascade.xml` ..... kaskádní klasifikátor (letadlo)
  - `control.py` ..... skript zajišťující otáčení kamery
  - `file.txt` ..... textový soubor s uloženými snímky
  - `mail.json` ..... nastavení pro odesílání e-mailů
  - `main.py` ..... detekční algoritmus
  - `pantilthat.txt` ..... záznam o poloze otáčecího modulu
  - `paperplane.xml` ..... kaskádní klasifikátor (papírové letadlo)
  - `settings.json` ..... nastavení parametrů detekce
  - `uploader.py` ..... skript zajišťující nahrávání na vzdálené úložiště